UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 4

Sistema de gestión de información para el apoyo al diagnóstico de pacientes intervenidos quirúrgicamente en el área abdominal

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Frank Delfín Reyes

Lázaro Edreniel Millares Bermúdez

Tutores: MSc. Ing. Rosalba Carralero Medina

Ing. Angel Alberto Vazquez Sánchez

Consultor: MSc. Dr. Leonardo Concepción Quiñonez

La Habana

Junio, 2014

AGRADECIMIENTOS

Lázaro:

Agradecer en especial a mis padres que sin ellos no hubiera sido posible esto, a toda mi familia que siempre ha estado ahí cuando la eh necesitado, a mis amigos, los que por un motivo u otro no están, a Rey que siempre me ha ayudado, a todos los amigos que he hecho en esta escuela, que me han enseñado mucho, a Maury, a Tito, a los Joses, a Frank, a Lorenzo, Dani, Marlon. A mi tutora Rosalba que se me olvido en la entrega anterior. En fin muchas gracias.

Frank:

Esta sección del trabajo de diploma ha sido la más complicada de redactar por muy fácil que parezca. En el desarrollo de esta investigación se contó con el apoyo de muchas personas a las cuales les quiero agradecer: Inicialmente agradezco a toda mi familia por confiar en mí, por apoyarme en mis decisiones y encaminarme en mis estudios. A mis padres por estar ahí en todo momento, por guiarme por el buen camino y ayudarme a convertirme en la persona que soy. A mis hermanos por ayudarme de una forma u otra durante estos años de estudio. A mi abuela. A mis amigos todos, a mis compañeros de clases que no influyeron directamente pero que siempre estuvieron presentes y que de alguna forma me apoyaron. A mi compañero de tesis el lachi (La lechuga asesina). A mis tutores. A mis profesores de los que he aprendido durante estos 5 años. A todos ustedes, gracias

Declaramos ser autores del presente trabajo “Sistema de gestión de información para el apoyo al diagnóstico de pacientes intervenidos quirúrgicamente en el área abdominal” y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ___ días del mes de _____ del año 2014.

_____	_____
Autor	Autor
Lázaro E. Millares Bermúdez	Frank Delfín Reyes
_____	_____
Tutor	Tutor
MSc. Ing. Rosalba Carralero Medina	Ing. Angel Alberto Vazquez Sánchez

Resumen

El presente trabajo se centra en el desarrollo de un sistema que facilite la gestión de los pacientes intervenidos quirúrgicamente en el área abdominal, en la unidad de cuidados intensivos del Hospital Enrique Cabrera. Este tiene como objetivo fundamental registrar, organizar, actualizar, conservar y mostrar la información de forma dinámica y segura, además de realizar el cálculo de los índices pronósticos.

Su desarrollo está basado en tecnologías libres o de código abierto y multiplataformas. Se utilizó PHP 5.4.16 como lenguaje de programación, MySQL 5.3.1 como gestor de base de datos, el framework Symfony 2.3.7 y como metodología de desarrollo XP.

El uso del sistema trae beneficios para el paciente ya que tendrá una Historia Clínica única y centralizada que garantiza el seguimiento y la seguridad de la información médica; por lo que posibilitará mayor rapidez y calidad en el servicio recibido. En cuanto al médico, este podrá acceder a la información clínica del paciente para facilitar los procesos de diagnóstico, tratamiento y seguimiento, y podrá realizar investigaciones médicas mediante la revisión de los diagnósticos.

Palabras claves: gestión de información, Índices pronósticos, paciente.

Índice

Índice de tablas

Índice de figuras

INTRODUCCIÓN

Las Tecnologías de la Información y las Comunicaciones (TIC), aparejados a la informática han alcanzado insertarse en todos los ámbitos de la vida diaria. Los sistemas automatizados se han convertido en un recurso de gran importancia, ya que el uso de estas modernas tecnologías garantiza procesar y obtener una mayor cantidad y variedad de información, seguridad, satisfacción al cliente, rapidez y que se pueda contar con datos actualizados en espacios de tiempo relativamente cortos.

Muchas son las ciencias que se apoyan en los sistemas automatizados para agilizar sus procesos. La medicina es una de estas y ha alcanzado logros con esta vinculación. Una de las causas de la inserción progresiva de las tecnologías informáticas en esta área es producto a que diariamente se manejan grandes volúmenes de datos y existe necesidad de obtener reportes estadísticos cada vez más específicos, además de mejorar los servicios de planificación sanitaria y de los recursos hospitalarios.

Otra de las mejoras de esta vinculación se puede observar en el aporte de nuevos conocimientos y habilidades en los especialistas para así reforzar y mejorar la toma de decisiones médicas y la atención al paciente. La combinación de las TIC y las técnicas para el procesamiento de datos permiten que las actividades médicas maximicen sus logros.

En los inicios de la integración fueron desarrollados sistemas médicos que surgieron solo para tareas administrativas, control de inventarios, admisión y alta de pacientes por mencionar algunas. Estos sistemas han evolucionado llegando a especializarse en los diferentes niveles de la salud, alcanzando un notable desarrollo en los hospitales, dando lugar al surgimiento de los Sistemas de Información Hospitalaria, o por sus siglas en inglés, HIS (Hospital Information System).

Un área en la medicina que cuenta con un alto desarrollo en la tecnología son los denominados procesos quirúrgicos; los cuales son realizados a una persona que va a ser expuesta a una cirugía. En esta área las complicaciones infecciosas relacionadas con intervenciones de cirugía abdominal han aumentado en los últimos años. La complejidad de estas, radica en el diagnóstico de las complicaciones quirúrgicas. Para ello se usan una serie de índices pronósticos que determinan la gravedad del paciente, facilitando así las diferentes decisiones a tomar por los especialistas.

Los índices pronósticos de mortalidad también ayudan a predecir los pacientes candidatos a presentar complicaciones serias o a morir. El trabajo con estos índices es muy engorroso y lento producto a la gran cantidad de variables que se deben tener en

cuenta para la puesta en práctica de los mismos. Lo que conlleva a que en ocasiones se demoren los diagnósticos que son de gran importancia para los pacientes.

Las principales deficiencias del proceso diagnóstico de pacientes intervenidos quirúrgicamente en el área abdominal se pueden resumir de la siguiente manera:

- La información del paciente obtenida luego de realizado un procedimiento quirúrgico es guardada en copia dura por lo que puede sufrir daños o pérdidas.
- El tener que analizar varios registros de datos del paciente demora el cálculo de los índices pronósticos con graves consecuencias para la salud del mismo.
- El cálculo de los índices pronóstico es realizado por los médicos de forma manual lo que puede introducir errores en la respuesta y el diagnóstico aplicado al paciente.
- La no existencia de un estándar de presentación de la información obtenida en estos índices dificulta su entendimiento.
- Existen pocas facilidades en la presentación de la información que se guarda, lo que dificulta los análisis estadísticos.

Por lo anterior se puede definir el siguiente **problema a resolver:** ¿Cómo contribuir al proceso de gestión de información y diagnóstico de pacientes intervenidos quirúrgicamente en el área abdominal?

Como **objeto de estudio**: la gestión de información y diagnóstico de pacientes intervenidos quirúrgicamente en el área abdominal. Como **campo de acción**: la gestión de información y diagnóstico de pacientes intervenidos quirúrgicamente en el área abdominal utilizando herramientas informáticas.

Para dar solución al problema planteado se ha definido como **objetivo general:** desarrollar un sistema de gestión de información que contribuya en el proceso de diagnóstico en pacientes intervenidos quirúrgicamente en el área abdominal.

Del cual se derivan los siguientes **objetivos específicos:**

- Determinar los referentes teóricos y metodológicos sobre el proceso de toma de decisiones de los especialistas en cirugía general ante pacientes intervenidos quirúrgicamente en el área abdominal.
- Diseñar e implementar un sistema informático que contribuya al proceso gestión de información y diagnóstico de pacientes intervenidos quirúrgicamente en el área abdominal.

- Validar la calidad del sistema informático de manera que cumpla con los aspectos funcionales y no funcionales determinados para el mismo.

Como **idea a defender** se plantea que, si se desarrolla un sistema que gestione la información y diagnóstico de pacientes intervenidos quirúrgicamente en el área abdominal, entonces se contribuirá al registro de los historiales, el cálculo de los índices pronósticos, la presentación de la información y los análisis estadísticos.

Se espera como **posibles resultados**:

Un sistema informático que contribuirá al registro de los historiales, el cálculo de los índices pronósticos, así como la presentación de la información y los análisis estadísticos.

Los **métodos científicos** empleados en la investigación son los siguientes:

Métodos teóricos:

• **Analítico-Sintético:** se utilizó en el proceso de análisis de la bibliografía utilizada, realizando una síntesis de la misma.

Método Empírico:

• **Observación:** se utilizó en la investigación para precisar el problema a resolver, las necesidades del cliente y los procesos que se llevan a cabo actualmente para el cálculo de los índices pronósticos.

Estructura capitular:

El documento está compuesto por 3 capítulos, de los cuales se expondrá a continuación una breve reseña.

Capítulo 1. Fundamentación Teórica: se expone la fundamentación teórica de la investigación, el mismo incluye: estudio de soluciones informáticas, las metodologías de desarrollo de software, tecnologías y herramientas necesarias para el desarrollo de la propuesta de solución.

Capítulo 2. Propuesta de solución: este apartado contiene lo referente a las fases de Exploración, Planeación y Diseño propuestas por XP, además los elementos fundamentales para la construcción de la propuesta como son los aspectos funcionales y requisitos no funcionales del sistema.

Capítulo 3. Implementación y Prueba: se reflejan los modelos que dan paso a la implementación propuestos por la metodología utilizada y las pruebas de software

realizadas. Contiene las tareas de ingeniería, los casos de pruebas de aceptación así como el análisis de los resultados de las pruebas realizadas al sistema.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se presentan los elementos teóricos que sirven de base a la investigación del problema planteado, se estudian los sistemas de información hospitalarios. Además, se fundamentan la metodología, las tecnologías y herramientas utilizadas en la construcción del sistema que se desea implementar y se realizan las valoraciones que justifican su elección.

1.2 Sistemas de información

La gestión de la información permite utilizar y adaptar las tecnologías de forma que aporten un valor real medible. Gestionar la información es tomar las decisiones estratégicas y tácticas para aportar valor que genere beneficio. La misma tiene que ver con la capacidad de la toma de decisiones, la supervisión y el control.

Un sistema de información (SI) se define entonces como un sistema capaz de recoger, almacenar y procesar datos para, después de un análisis, obtener información útil y necesaria para la organización en la que está inmerso. Su diseño incluye analizar datos para producir información útil que contribuya a la toma de decisiones, estos datos son almacenados de manera estructurada en bases de datos para consultas y estudios posteriores. (Cotos Yáñez, et al., 2005)

Un sistema de información comprende toda la cadena de operaciones que comienza en la observación y recolección de datos, pasando por su almacenamiento y análisis, hasta el uso efectivo de la información derivada en un proceso de toma de decisiones. La aceptación de un sistema de información supone beneficios que pueden dividirse en las siguientes categorías (Cotos Yáñez, et al., 2005) :

- Beneficios de rendimiento.
- Beneficios de eficacia.
- Beneficios de ventaja competitiva.

Un SI puede considerarse formado por distintos componentes (Cotos Yáñez, et al., 2005):

- Los datos que son las entradas introducidas en el sistema, necesarias para generar información.
- El usuario, la persona que interactúa con el SI, quien lo alimenta de datos, maneja la información y/o utiliza los resultados generados.

- El hardware, necesario para que el sistema de información pueda comunicarse, procesar y almacenar información.
- Los procedimientos, que se ejecutan sobre los datos y que producen diferentes tipos de resultados. Forman parte del software del sistema de información y su objetivo es que las entradas sean procesadas correctamente y generen los resultados esperados.

Todos estos componentes colaboran para que los SI puedan desempeñar sus actividades básicas, de entrada, almacenamiento, procesado y salida de información. (Cotos Yáñez, et al., 2005)

Luego de definidos los SI se puede concluir que, dadas las características de los mismos, son aplicables a ramas de la ciencia donde sea necesaria la realización de las funciones que permiten la gestión continua de datos, como lo son la medicina, la gestión empresarial, la gestión medioambiental, entre otras.

1.2.1 Sistema de información hospitalario

Un SI es un conjunto de instrucciones organizadas, sistematizadas y lógicas que se relacionan entre sí por medio de un lenguaje informático con el fin de obtener información, analizarla, relacionarla y generar nueva información para satisfacer las necesidades de las áreas administrativas, operativas de una organización en general.

Un Sistema de Información Hospitalario (HIS, por sus siglas en inglés) está orientado a satisfacer las necesidades de generación de información, para almacenar, procesar y reinterpretar datos médico- administrativos de cualquier institución hospitalaria. Así como a la optimización de los recursos humanos y materiales.

Todo HIS genera reportes e informes en dependencia del área o servicio para el cual se requiera, dando lugar a la retroalimentación de la calidad de la atención de los servicios de salud. La principal función de un HIS es la de apoyar las actividades en los niveles operativos, tácticos y estratégicos dentro de un Hospital. (Medicina, 2003)

1.3 Índices pronósticos de mortalidad

Los índices pronósticos de mortalidad son un instrumento utilizado en la estimación de la evolución clínica y pronóstico de los pacientes ingresados en los servicios y áreas que atienden al paciente crítico. Secundariamente, se han aplicado en la evaluación de la efectividad y eficiencia de los servicios de medicina intensiva. La gran mayoría de estos sistemas establecen el riesgo de mortalidad del paciente crítico basándose en los

valores obtenidos tras asignar una determinada puntuación a distintas variables demográficas, fisiopatológicas y clínicas. (López Álvarez, et al., 2001)

Uno de los padecimientos al que se le es aplicado estos índices, es la peritonitis. La peritonitis es una inflamación del peritoneo, la membrana serosa que recubre parte de la cavidad abdominal y las vísceras. La peritonitis puede ser localizada o generalizada, y puede resultar de la infección (a menudo debido a la ruptura de un órgano hueco, como puede ocurrir en el traumatismo abdominal o apendicitis) o de un proceso no infeccioso.

La peritonitis secundaria a perforación del colon conlleva mortalidad y morbilidad elevadas. La identificación de factores de riesgo de mortalidad postoperatoria ha llevado a la elaboración de sistemas de puntuación con significado pronóstico que han sido aplicados a diferentes escenarios clínicos y alguno ha sido diseñado específicamente para pacientes quirúrgicos.

Los índices pronósticos permiten cuantificar y predecir el riesgo de morbilidad y mortalidad según parámetros fisiológicos, analíticos o clínicos, y su aplicación es una forma válida y rigurosa para medir la probabilidad de complicaciones y mortalidad postoperatoria.

El uso de un sistema de puntuación que pueda proporcionar una estimación objetiva del riesgo individual de mortalidad postquirúrgica del paciente es una gran ayuda para una correcta planificación de la estrategia terapéutica y para la gestión de recursos sanitarios, principalmente ante enfermos que requieren estancia y tratamiento en unidades de cuidados intensivos. (Domenico Fraccalvieri, 2009)

La predicción del riesgo de muerte en la peritonitis a través de diferentes índices pronósticos como: Acute Physiological and Chronic Health Evaluation II (APACHE II), Índice Peritonítico de Mannheimer (IPM), Índice de Defunción Orgánica Múltiple (IDOM), el Estado Fisiológico Agudo Simplificado II (SAPS II, siglas en inglés), permiten la estimación de la severidad de la enfermedad y del desenlace final, los cuales constituyen un arma importante del cirujano y el intensivista en el momento de evaluar, monitorizar, y planear las intervenciones terapéuticas, lo que pudiera repercutir positivamente en la supervivencia de los enfermos.

1.4 Soluciones similares

Con el objetivo nutrir la investigación con respecto a las tendencias de desarrollo de este tipo de sistemas y las funcionalidades básicas que deben poseer se analizan las siguientes soluciones:

1.4.1 Sushrut

C-CAD (Centre for development of advanced computing) Sushrut, es un HIS desarrollado en la India con el objetivo de racionalizar el flujo de tratamiento de un paciente en el hospital. Presenta una arquitectura cliente servidor de base de datos.

El sistema de Operación contiene información sobre la disponibilidad de todos los quirófanos, equipo y herramientas. La planificación de operaciones es la función principal de este sistema a través de la cual se puede aprobar, cancelar o reprogramar la operación. Las actividades programadas han de ser validadas por los cirujanos principales. Permite la entrada y validación de un registro detallado de la operación, la anestesia y el mantenimiento del post-operatorio. (SUSHRT, 2006)

1.4.2 Care2x

HIS alemán que integra datos, funciones y flujo de tareas en un entorno de cuidados de la salud. Presenta una arquitectura cliente servidor. El sistema contiene el sistema quirófanos que se encarga de las siguientes funcionalidades de las salas de operaciones (Care2x, 2002):

- Documentar los procedimientos quirúrgicos (cirugía, anestesia, enfermería, materiales y medicinas).
- Planificador de actividades del quirófano.
- Funciones de búsqueda y archivo.
- Clasificación internacional de enfermedades.
- Planificador de operaciones quirúrgicas.

1.4.3 GoWin

Es un software español desarrollado por Valen Computer que está compuesto por todos y cada uno de los sistemas necesarios para una gestión sanitaria integral, enlazando las áreas asistenciales con las administrativas para conseguir la mejor eficiencia en la gestión de los centros.

GoWin Qui es un sistema de información del quirófano que ha sido desarrollado e integrado en el HIS, para poder gestionar todas las funciones que ha de realizar el personal de quirófano, empezando con la hoja de trabajo, pasando por la citación, las validaciones y acabando en la hoja quirúrgica, que generará los informes correspondientes.

Sistemas de Gowin Qui - Sistema de información del quirófano:

Sistema de citación y planificación: el sistema de citación y planificación es el encargado de gestionar los quirófanos, su definición y la programación realizada en los quirófanos mencionados. El sistema permite la gestión de la programación de cada quirófano. Entre muchas otras funciones, el sistema permite la creación, modificación, exclusión, validación y la reprogramación, según la lista de espera.

Sistema de hoja quirúrgica: el sistema de la hoja quirúrgica permite almacenar todos los datos que hacen referencia a la intervención quirúrgica, la codificación, registro de enfermería y registro de prótesis.

Sistema de parametrización: el sistema de parametrización permite la configuración del sistema. Se definen las tablas necesarias y se configura el sistema para su funcionamiento según las especificaciones de cada organización. (S.A, 2008)

1.4.4 Valoración del estudio de las soluciones similares

Los HIS mencionados, en su mayoría no son multiplataforma y operan en sistemas operativos privativos lo que resulta ser un inconveniente para insertarse en el mercado del software. Además, se conoce que ninguno de los tres sistemas analizados son soluciones libres, condición que limita al país la adquisición de los mismos y las futuras mejoras a las funcionalidades que brindan, debido a las restricciones y alto coste de sus licencias.

Gran parte de estas soluciones representan al área quirúrgica como un sistema que se integra con las demás áreas del hospital, lo que permite una mejor atención al paciente, aunque existen casos de sistemas que solo se encargan de la gestión de información en esta área. En su mayoría se podría decir que son aplicaciones web que se basan en una arquitectura cliente-servidor, lo que ratifica que esta es una tendencia en el desarrollo de estos sistemas aunque hay algunas como GoWin, que son aplicaciones de escritorio lo que dificultaría su despliegue y soporte.

Como estos sistemas no cumplen con los objetivos de la investigación ya que no realizan el cálculo de los índices pronósticos, se propone la implementación de un sistema que registre toda la información de los pacientes, realice el cálculo de dichos índices, controle la evolución de los pacientes y dé un seguimiento a las políticas de antibióticos. Dicho sistema será implementado para la Web, aprovechando la rapidez y poco consumo de recursos que esta brinda.

1.5 Metodologías de Desarrollo

Para guiar el proceso de desarrollo de un software es imprescindible el uso de una metodología, que sea capaz de guiar al equipo de trabajo con el objetivo de aumentar

la calidad y eficiencia en el proceso de creación de software. Estas metodologías de desarrollo de software son un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software. En la actualidad la cantidad y diversidad de metodologías de desarrollo de software han aumentado en gran medida.

Dentro de las metodologías de desarrollo existen dos grandes grupos: las metodologías tradicionales o robustas y las metodologías ágiles o ligeras. Las primeras se enfatizan en el uso exhaustivo de documentación durante todo el ciclo de vida del proyecto, se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Además, estas metodologías son recomendadas para proyectos con grandes equipos de desarrollo. Mientras que las ágiles están orientadas a proyectos pequeños con requisitos muy cambiantes donde se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad. También dan mayor valor al individuo, a la colaboración entre el cliente y el equipo de desarrollo y al desarrollo incremental del software con iteraciones muy cortas. (Letelier, y otros, 2009)

Entre las metodologías que guían el desarrollo de un software se encuentran las metodologías ágiles Programación Extrema (XP), SCRUM y Desarrollo Basado en Funcionalidades (FDD por sus siglas en inglés Feature Driven Development) y las tradicionales Proceso Unificado de Desarrollo (RUP por sus siglas en inglés Rational Unified Process) y Microsoft Solution Framework (MSF).

Selección de la Metodología a utilizar

Se decide utilizar XP porque es una metodología que se adecua perfectamente a las características que presenta el proyecto como son: requisitos muy cambiantes, existe un alto riesgo técnico, posee un equipo pequeño y poco tiempo de desarrollo. Además, el cliente forma parte del equipo de desarrollo, lo que permite establecer un vínculo de comunicación entre este y los desarrolladores, elevando así la calidad del sistema a desarrollar.

Esta metodología fue desarrollada por Kent Beck, el cual plantea lo siguiente: «Todo en el software cambia. Los requisitos cambian. El diseño cambia. El negocio cambia. La tecnología cambia. El equipo cambia. Los miembros del equipo cambian. El problema no es el cambio en sí mismo, puesto que sabemos que el cambio va a suceder; el

problema es la incapacidad de adaptarnos a dicho cambio cuando éste tiene lugar.» (Fernández Escribano, 2002)

XP surge como respuesta y posible solución a los problemas derivados del cambio en los requerimientos, se plantea como una metodología a emplear en proyectos de riesgo y aumenta la productividad. Tiene como objetivo: la satisfacción del cliente. Esta metodología trata de dar al cliente el software que él necesita y cuando lo necesita. Por tanto, debe responder muy rápido a las necesidades del cliente, incluso cuando los cambios sean al final de ciclo de la programación. (González Barbone, 2007)

Otro objetivo es potenciar al máximo el trabajo en grupo. Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del software. XP define cuatro variables para proyectos de software: coste, tiempo, calidad y ámbito. (Calero Solís, 2003)

1.6 Herramienta de Modelado

El desarrollo del software ha generado la construcción de herramientas para que los analistas y diseñadores de software realicen los procesos de modelado del sistema de forma eficiente, con mayor calidad y fiabilidad.

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son un ejemplo de ello, constituyen un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software.

Actualmente existen una gran cantidad de herramientas CASE, entre las que se encuentran Rational Rose, Enterprise Architect y Visual Paradigm for UML.

A continuación se hace un estudio de las siguientes herramientas:

Visual Paradigm

Herramienta profesional que brinda soporte al modelado visual mediante la notación del Lenguaje Unificado de Modelado (UML). Soporta la totalidad del ciclo vital del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Herramienta con funcionalidades muy completas y con abundantes facilidades, su empleo contribuye a la construcción rápida de aplicaciones de calidad, óptimas y con una minoría en cuanto a costo. Permite además modelar todo tipo de diagramas de clases, código inverso, generar el código desde los diagramas, posee interoperabilidad con otras aplicaciones e integración con distintos Entornos de Desarrollo Integrado,

suministra abundantes tutoriales y crea de manera simple toda la documentación, incluyendo formatos tales como PDF y HTML.

Algunas de las ventajas que posee esta herramienta son las siguientes:

- Creado para múltiples plataformas.
- Ofrece la oportunidad de intercambiar información mediante la importación y exportación de ficheros con aplicaciones.
- Brinda un apoyo adicional en cuanto a generación de artefactos automáticamente.
- Se puede generar código a partir de los diagramas, así como obtener los diagramas a partir del código, denominado generación de código e ingeniería inversa.
- Posee las funcionalidades para documentar todo el trabajo sin necesidad de utilizar herramientas externas.

Después de modelado el sistema ya todo estaría listo para su implementación, es ahí donde vienen a jugar su papel la selección de uno o varios lenguajes de programación adecuados en correspondencia con la aplicación que se quiera construir. (SunMicrosystem, 2008)

Rational Rose

Rational Rose es la herramienta CASE desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases entregables.

Es una herramienta con plataforma independiente que ayuda a la comunicación entre los miembros de equipo, a monitorear el tiempo de desarrollo y a entender el entorno de los sistemas. Una de las grandes ventajas que posee es que utiliza la notación estándar en la arquitectura de software UML. Esta permite a los arquitectos de software y desarrolladores visualizar el sistema completo utilizando un lenguaje común, además los diseñadores pueden modelar sus componentes e interfaces en forma individual y luego unirlos con otros componentes del proyecto.

Rational Rose permite completar una gran parte de flujos fundamentales en concreto (IBM, 2009):

- Modelado del negocio.

- Captura de requisitos.
- Análisis y diseño.
- Implementación.
- Control de cambios y gestión de configuración.

Enterprise Architect

Enterprise Architect (EA) es una herramienta comprensible de diseño y análisis UML, cubriendo el desarrollo de software desde el paso de los requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. Es una herramienta multi-usuario, basada en Windows, diseñada para ayudar a construir software robusto y fácil de mantener. Ofrece salida de documentación flexible y de alta calidad.

Algunas de las principales características de EA son las siguientes (System, 2008):

- Velocidad, estabilidad y rendimiento:

UML provee beneficios significativos para ayudar a construir modelos de sistemas de software rigurosos y donde es posible mantener la trazabilidad de manera consistente. Soporta este proceso en un ambiente fácil de usar, rápido y flexible.

- Trazabilidad de extremo a extremo:

Provee trazabilidad completa desde el análisis de requerimientos hasta los artefactos de análisis y diseño, a través de la implementación y el despliegue. Combinados con la ubicación de recursos y tareas incorporados, los equipos de administradores de proyectos y calidad están equipados con la información que ellos necesitan para ayudarles a entregar proyectos en tiempo.

- Construido sobre UML 2.1:

Las bases de EA están construidas sobre la especificación de UML 2.0, pero no se detiene ahí, usa perfiles UML para extender el dominio de modelado, mientras que la validación del modelo asegura integridad.

Además soporta generación e ingeniería inversa de código fuente para muchos lenguajes populares, incluyendo C++, C#, Java, Delphi, VB.Net, Visual Basic y PHP. Con un editor de código fuente con "resaltador de sintaxis" incorporado, EA le permite navegar y explorar su modelo de código fuente en el mismo ambiente. Las plantillas de generación de código le permiten personalizar el código fuente generado a las especificaciones de su compañía.

Selección de la herramienta de modelado a utilizar

Analizadas estas herramientas se decide utilizar Visual Paradigm 8.0 ya que genera toda la documentación del ciclo de vida del software cumpliendo con los estándares establecido. Otra de las características por la que se decide usar, es su disponibilidad en múltiples plataformas, ya que no obliga al usuario a desarrollar solo en el sistema operativo Windows, sino que está disponible en sistemas operativos como Linux y Unix. Además, se integra con Netbeans.

1.7 Lenguajes de Programación Web

Para utilizar un lenguaje de programación Web, se debe conocer con exactitud cuáles son las necesidades de desarrollo y si el lenguaje en cuestión satisface las mismas. Los lenguajes de programación para la Web se dividen en dos grupos a tono con la propia arquitectura cliente-servidor: los lenguajes del lado del Cliente y los lenguajes del lado del Servidor.

Los lenguajes del lado del Cliente, entre los que se encuentran HTML, Twig, JavaScript y CSS, son independientes del servidor lo cual significa que pueden ser “digeridos” directamente por el servidor y no necesitan pre-procesamiento. Entre los lenguajes del lado del Servidor, los cuales se caracterizan por desarrollar la lógica de negocio dentro del servidor además de encargarse del acceso a bases de datos y al tratamiento de la información, se distinguen Python, Perl y PHP.

Lenguajes del lado del Cliente

HTML

HTML es un lenguaje de programación que se utiliza para el desarrollo de páginas de Internet. Se trata de la sigla de HyperText Markup Language, es decir, Lenguaje de Marcas de Hipertexto.

Permite describir la estructura y el contenido en forma de texto, además de complementar el texto con objetos tales como imágenes. Este lenguaje se escribe mediante etiquetas, que aparecen especificadas por corchetes angulares. Por otra parte, HTML permite incluir scripts (por ejemplo, de JavaScript), códigos que pueden modificar el comportamiento de los navegadores web y de otros procesadores de HTML. Es el lenguaje de marcado predominante para la elaboración de páginas web. (Vega, et al., 2011)

JavaScript

Se trata del lenguaje del lado del cliente más utilizado gracias a su compatibilidad con la mayoría de los navegadores. Mediante su uso se pueden crear efectos especiales en las páginas y definir interactividades con los usuarios. Gran parte de la programación en este lenguaje se dirige a escribir funciones que responden a determinados eventos como el movimiento del mouse, la utilización de teclas, la carga de páginas, entre otros. También es muy utilizado a la hora de validar los datos introducidos por los usuarios a través de los formularios. Es necesario precisar que existen dos tipos de JavaScript: por un lado está el que se ejecuta en el cliente, este es el JavaScript propiamente dicho, aunque técnicamente se denomina Navigator JavaScript y por el otro el que se ejecuta en el servidor, más reciente y denominado LiveWire JavaScript. (Eguiluz Perez, 2010)

CSS (Cascade Style Sheets)

CSS es un lenguaje del lado del cliente creado para describir cómo se mostrará un documento, definido con HTML, en pantalla o cómo se va a imprimir e incluso cómo será pronunciada la información presente en el documento a través de un dispositivo de lectura. Mientras que el lenguaje HTML se utiliza para marcar los contenidos, es decir, para designar lo que es un párrafo, lo que es un titular o lo que es una lista de elementos, el lenguaje CSS se utiliza para definir el aspecto de todos los contenidos, es decir, el color, tamaño y tipo de letra de los párrafos de texto, la separación entre titulares y párrafos, la tabulación con la que se muestran los elementos de una lista, etc. CSS permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento. (Eguiluz, 2009)

Lenguajes del lado del Servidor

Python

Es un lenguaje de programación, manejado como proyecto de software libre, de propósito general, o sea, permite la creación de todo tipo de aplicaciones incluyendo los sitios web. Habitualmente se le compara con Perl y sus usuarios consideran que es más limpio para programar, aunque esto no es más que un punto de vista de los mismos. Es un lenguaje multiplataforma y multiparadigma, esto último permite a los programadores adoptar un estilo de programación particular. Es visto como un lenguaje exitoso debido a su facilidad de aprendizaje, su orientación a programadores promedio y la limpieza de su código. (Angel Alvarez, 2003)

Perl

Perl es un acrónimo de Practical Extracting and Reporting Language, lo cual se traduce en que se trata de un lenguaje práctico para extraer información de archivos de texto y generar informes a partir del contenido de los ficheros. Es un lenguaje libre asociado a la plataforma Unix, esto no indica que no esté disponible en otros Sistemas Operativos como Windows. Al ser Perl un lenguaje de programación interpretado el código contenido en sus scripts no se compila sino que cada vez que se quiere ejecutar se lee el código y se pone en marcha interpretando lo que hay escrito. (Angel Alvarez, 2001)

PHP

Es un lenguaje de programación interpretado usado generalmente para la creación de contenido para sitios o aplicaciones Web. Sus siglas representan un acrónimo recursivo que significa “Hypertext Preprocessor” aunque vale destacar que inicialmente se denominó Personal Home Page. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl a lo cual se incorporan características específicas. Es un producto de software libre, debido a esto cuenta con la colaboración de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y se reparen rápidamente.

Debido al intenso trabajo de la extensa comunidad de desarrolladores que posee, PHP se actualiza continuamente con mejoras y extensiones de lenguaje para ampliar sus capacidades, manteniendo de igual manera una amplia, práctica y actualizada documentación.

PHP ha sido diseñado de forma muy modular y el surgimiento de nuevas librerías contribuye a la notable sencillez que posee. Toda esta funcionalidad está basada en librerías que en su mayor parte no han sido desarrolladas por el equipo de PHP. (PHP.net, 2010)

Entre sus características fundamentales están:

- Libre y Gratuito: al tratarse de software libre puede descargarse y utilizarse en cualquier aplicación, personal o profesional, de manera completamente libre.
- Rapidez de ejecución.
- Mantiene un bajo consumo de recursos de máquina.
- Gran seguridad: muy poca probabilidad de corromper los datos.
- Permite las técnicas de Programación Orientada a Objetos.
- No requiere definición de tipos de variables.

- Gran popularidad: existe una gran comunidad de programadores y desarrolladores que continuamente implementan mejoras en su código, y que se ayudan entre sí cuando se enfrentan con un problema. Posee una amplia documentación en Internet, incluyendo una gran variedad de ejemplos y de ayudas.
- Eficiencia: con escaso mantenimiento puede soportar sin problemas millones de visitas diarias.
- Sencilla integración con múltiples bases de datos: esencial para una Página Web verdaderamente dinámica es una correcta integración con bases de datos, trabaja con bases de datos muy conocidas como MySQL, PostgreSQL y Oracle.
- Versatilidad: puede utilizarse con la mayoría de sistemas operativos como Unix (Linux, Solares, FreeBSD, y otros), como con Windows.
- Gran número de funciones predefinidas: a diferencia de otros lenguajes de programación, fue diseñado para el desarrollo de páginas web dinámicas, por ello está dotado de un gran número de funciones que posibilitan la simplificación de códigos complejos o poco complejos que se reducen a una palabra.

La versión 5 de PHP provee a los desarrolladores de un magnífico trabajo con el Paradigma Orientado a Objetos que permite la reutilización de código entre otras facilidades, de este modo se coloca más a tono con las exigencias de la programación moderna. (PHP.net, 2010)

Selección de los Lenguajes de Programación Web

Luego de realizado el análisis de los lenguajes utilizados en la actualidad para el desarrollo de aplicaciones Web, y atendiendo a que los mismos están divididos en Lenguajes del lado del Cliente y Lenguajes del lado del Servidor, se escogen:

Lenguajes del lado del Cliente

La combinación de los lenguajes HTML, JavaScript y CSS es la más utilizada en la actualidad para desarrollar el contenido referente a la parte del cliente en las aplicaciones Web debido a que los tres se complementan, brindando una amplia gama de posibilidades a los desarrolladores.

HTML como lenguaje principal con el objetivo de describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes, sonido, etc.

JavaScript sigue siendo uno de los lenguajes imprescindibles a la hora de crear interacciones con el usuario mediante el manejo de eventos. Actualmente con el surgimiento del AJAX, las prestaciones que brinda se han elevado considerablemente.

Si se quiere controlar el estilo y formato de múltiples páginas web de manera simultánea, separando el estilo de la presentación de los documentos HTML, el uso de CSS se hace necesario.

Por tanto, se seleccionan los lenguajes HTML 5, JavaScript y CSS 3 como lenguajes de programación del lado del Cliente atendiendo, además, a la experiencia que se tiene sobre su uso.

Lenguajes del lado del Servidor

Luego del análisis realizado anteriormente se llegó a la conclusión de que Perl, Python y PHP, son lenguajes de programación, pero entre ellos, PHP se adecua mejor a las exigencias de la investigación, además, a diferencia de los otros dos lenguajes mencionados, fue diseñado desde cero con el objetivo de desarrollar aplicaciones Web.

Las tareas más habituales para la realización de estos tipos de aplicaciones pueden hacerse con PHP de forma fácil, rápida y efectiva. La curva de aprendizaje de PHP no es elevada por cuanto los resultados de su uso son rápidamente observables. PHP no obliga al uso de determinada metodología al programar, como otros lenguajes tampoco, sin embargo, los desarrolladores son libres de elegir para su trabajo cualquier técnica que les permita mantener su código ordenado. (PHP.net, 2010)

Por lo anterior, se selecciona el lenguaje PHP 5.4.16 como lenguaje de programación del lado del Servidor.

1.8 Framework

Los framework proporcionan una estructura o marco para desarrollar proyectos. Simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Facilita la programación de aplicaciones, porque encapsula operaciones complejas en instrucciones sencillas. En resumen, poseen una serie de clases, funciones o librerías, que junto a convenciones comunes, se organizan bajo una estructura sobre la cual se desarrollan las aplicaciones (ElWebMaster, 2009).

Entre las ventajas de usar framework para desarrollar la solución, se pueden contar (ElWebMaster, 2009):

- No hay que preocuparse por mantener actualizadas las distintas partes, generalmente los frameworks son abiertos y soportados por una comunidad que actualizan la funcionalidad y corrigen los bugs de manera sostenida.
- Aprovechan los componentes existentes aumentando la velocidad de desarrollo.
- Reducción en el tiempo de desarrollo de nuevas aplicaciones.
- Reducción del costo de mantenimiento.
- Mayor nivel de confiabilidad (comparado con escribir código nuevo), en la medida que hay reuso y el framework se estabiliza.
- Abstracción de URLs (Localizador de Recursos Uniforme) y sesiones, porque no es necesario manipular directamente las URLs ni las sesiones, el framework ya se encarga de hacerlo.
- Fácil acceso a datos. Incluyen las herramientas e interfaces necesarias para integrarse con herramientas de acceso a datos, en bases de datos, XML, etc.
- Autenticación y control de acceso, pues incluyen mecanismos para la identificación de usuarios mediante login y password que permiten restringir el acceso a determinadas páginas a determinados usuarios.

Como desventajas se pueden nombrar (ElWebMaster, 2009):

- Agrega código adicional que no es elaborado por el programador de la aplicación.
- Hay que invertir tiempo en aprender a usarlos.
- En algunos casos una aplicación desarrollada con un framework puede ser más lenta (en cuanto a rendimiento) que una diseñada y desarrollada sin usar el mismo.

Cakephp

Cakephp provee una base robusta para las aplicaciones web. Puede manejar cualquier aspecto, desde la solicitud inicial del usuario hasta el renderizado final de la página web. Además, como éste sigue los principios Modelo-Vista-Controlador, permite fácilmente personalizar y extender muchos aspectos de ésta.

El framework también suministra una estructura de organización básica, desde los nombres de los archivos hasta los de las tablas de la base de datos, manteniendo toda la aplicación consistente y lógica. (CakePHP, 2008)

Cakephp permite:

- Internacionalización y localización.
- Validación de datos.
- Limpieza de Datos.
- Manejo de errores.
- Depuración.
- Paginación.

Está caracterizado por (CakePHP, 2008):

- Compatible con PHP4 y PHP5.
- CRUD de la base de datos integrado.
- URLs amigables.
- Sistema de plantillas rápido y flexible.
- Ayudas para AJAX, Java script, HTML, forms.
- Trabaja en cualquier subdirectorio del sitio.
- Scaffolding (andamiaje) de las aplicaciones.
- Listas de Control de Acceso.
- Componentes de seguridad y sesión.

Zend Framework

Zend Framework es un framework de código abierto y orientado a objetos para facilitar el desarrollo de aplicaciones web con PHP 5, la implementación del código es 100% orientada a objetos. A menudo, es considerado una biblioteca de componentes, debido a que estos poseen bajo acoplamiento entre sí, lo cual permite reutilizarlos con un alto grado de independencia. Proporciona además una sofisticada implementación del patrón Modelo-Vista-Controlador, el cual puede ser utilizado para fijar la estructura básica de las aplicaciones desarrolladas con el framework. (Framework, 2009)

Entre sus principales características se pueden mencionar (Framework, 2009):

- Trabaja con Modelo-Vista-Controlador.

- El Marco de Zend también incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple para consultar su base de datos, sin tener que escribir ninguna consulta SQL.
- Una solución para el acceso a base de datos que balancea el Mapeo Relacional de Objetos con eficiencia y simplicidad.
- Completa documentación y pruebas de alta calidad.
- Clientes para servicios web, incluidos Google Data APIs y StrikeIron.
- Abstracción de la base de datos fácil de usar.
- Componentes que implementan formas de representación de formularios HTML.
- Validación y filtrado de datos.
- Servicios de autenticación y autorización de usuario.

CodeIgniter

CodeIgniter contiene una serie de librerías que sirven para el desarrollo de aplicaciones web y además propone una manera de desarrollarlas que se deberá seguir para obtener provecho de la aplicación. Este marca una manera específica de codificar las páginas web y clasificar sus diferentes scripts, que sirve para que el código esté organizado y sea más fácil de crear y mantener. CodeIgniter implementa el patrón arquitectónico Modelo-Vista-Controlador, que es un estándar de programación de aplicaciones, utilizado tanto para hacer sitios web como programas tradicionales.

Entre las características más importantes se pueden mencionar (Alvarez, 2009):

- Compatibilidad: es compatible con la versión PHP 4, lo que hace que se pueda utilizar en cualquier servidor, incluso en algunos antiguos, también en PHP 5.
- Facilidad de instalación: solo es necesaria una cuenta de FTP para subir CodeIgniter al servidor y su configuración se realiza con la edición de un archivo.
- Flexibilidad: define una manera de trabajar específica, pero en muchos de los casos podemos seguirla o no, y sus reglas de codificación muchas veces se pueden saltar para trabajar más a gusto. Algunos sistemas como el uso de plantillas son totalmente opcionales.
- Ligereza: el núcleo de CodeIgniter es bastante ligero, lo que permite que el servidor no se sobrecargue interpretando o ejecutando grandes porciones de código.

- Documentación tutorializada: la documentación de este es fácil de seguir y de asimilar, porque está escrita en modo de tutorial, aunque no facilita mucho la referencia rápida, cuando ya se sabe acerca del framework y se quiere consultar sobre una función o un método en concreto.

Symfony

Symfony es un framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación del sistema. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Es el framework más documentado, y cuenta con páginas de documentación distribuidas en libros gratuitos y tutoriales. (Symfony.es, 2010)

Symfony está caracterizado por (Symfony.es, 2010):

- Fácil de instalar y configurar en sistemas Windows, Mac y Linux.
- Funciona con todas las bases de datos comunes (MySQL, PostgreSQL, Oracle, SQL Server de Microsoft).
- Compatible solamente con PHP 5, para asegurar el mayor rendimiento y acceso a las características más avanzadas de PHP.
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
- Preparado para aplicaciones empresariales, porque se puede adaptar con facilidad a las políticas y arquitecturas propias de cada empresa u organización.
- Traducido a más de 40 idiomas y fácilmente traducible a cualquier otro idioma.

Symfony automatiza la mayoría de elementos comunes de los proyectos web, como por ejemplo (Symfony.es, 2010):

- La capa de internacionalización que incluye Symfony permite la traducción de los datos y de la interfaz, así como la adaptación local de los contenidos.

- Los formularios incluyen validación automatizada y relleno automático de datos, lo que asegura la obtención de datos correctos y mejora la experiencia del usuario.
- Los datos incluyen mecanismos de escape que permiten una mejor protección contra los ataques producidos por datos corruptos.
- La gestión de la caché reduce el ancho de banda utilizado y la carga del servidor.
- La autenticación y la gestión de credenciales simplifican la creación de secciones restringidas y la gestión de la seguridad de usuario.
- El sistema de enrutamiento y las URL limpias permiten considerar a las direcciones de las páginas como parte de la interfaz, además de estar optimizadas para los buscadores.
- El soporte de e-mail incluido y la gestión de APIs permiten a las aplicaciones web interactuar más allá de los navegadores.
- Los listados son más fáciles de utilizar debido a la paginación automatizada, el filtrado y la ordenación.

Selección del framework a utilizar

Tanto Symfony, Zend Framework, CakePHP y CodeIgniter son frameworks PHP5 con gran éxito entre los desarrolladores, sin embargo, es necesario decidir con cuál de ellos trabajar.

Dos de las ventajas más importantes con las que cuenta Symfony son la calidad de su código fuente y la gran cantidad de documentación disponible. Además, Symfony es una herramienta para el desarrollo de aplicaciones rápidas y se adapta fácilmente a los cambios que puedan surgir durante el desarrollo de software. Por tanto, se selecciona Symfony como framework PHP atendiendo, además, a las siguientes ventajas:

- Facilita herramientas para desarrollar aplicaciones Web de alta complejidad.
- Extensible a través de multitud de bundles disponibles para su descarga e instalación.
- El código desarrollado es más fácil de mantener.
- Implementación, de manera bien creativa, del patrón de diseño Modelo -Vista-Controlador.
- Utilización de otros patrones de diseño como Separación en Capas.

- Existencia de tutoriales y ejemplos de código disponible tanto en Internet como en la UCI.

1.9 Entorno de Desarrollo Integrado (IDE)

El IDE es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien, puede utilizarse para varios. Es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Delphi, Visual Basic, etc. (JELSOFT, 2008)

Entre los IDE's estudiados se encuentran los que a continuación se mencionan.

Eclipse

Eclipse es un IDE, multiplataforma, abierto y extensible. Eclipse es Java, aunque da soporte a otros lenguajes de programación, como son: C/C++, Cobol, Fortran, PHP o Pyton. Este entorno de desarrollo fue creado inicialmente por la IBM y actualmente es desarrollado por la Fundación Eclipse.

Como características principales presenta:

- Ha sido diseñado de forma que pueda ejecutarse en cualquier plataforma. La última versión estable se encuentra disponible para los sistemas operativos Windows, Linux, Solaris, AIX, HP-UX y Mac OSX.
- Emplea un diseño basado en sistemas (plug-in) los cuales se le pueden añadir para extender sus funcionalidades.
- Todas las versiones de Eclipse necesitan tener instalado en el sistema una máquina virtual Java (JVM), preferiblemente JRE (Java Runtime Environment) o JDK (Java Developer Kit) de Sun.
- Eclipse se distribuye bajo licencia EPL (Eclipse Public License). Esta licencia es considerada como libre. La licencia EPL permite usar, modificar, copiar y distribuir nuevas versiones del producto licenciado. El antecesor de EPL es CPL (Common Public Licencse) escrita por IBM. (eclipse.org, 2008)

Zend Studio

Es un completo IDE para el lenguaje de programación PHP. Está escrito en Java, y está disponible para las plataformas Microsoft Windows, Mac OS X y GNU/Linux. Además, sirve de editor de texto para páginas PHP, proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código.

Algunas de las características que posee son (Álvarez, 2007):

- No requiere la instalación previa de PHP ni del entorno de ejecución de Java.
- Soporte para PHP 4 y PHP 5.
- Resaltado de sintaxis, autocompletado de código, ayuda de código y lista de parámetros de funciones y métodos de clase.
- Plegado de código (comentarios, bloques de phpDoc, cuerpo de funciones y métodos e implementación de clases).
- Inserción automática de paréntesis y corchetes de cierre.
- Sangrado automático y otras ayudas de formato de código.
- Emparejamiento de paréntesis y corchetes.
- Detección de errores de sintaxis en tiempo real.
- Funciones de depuración: botón de ejecución y traza, marcadores, puntos de parada (breakpoints), seguimiento de variables y mensajes de error del intérprete de PHP. Permite también la depuración en servidores remotos (requiere Zend Platform).
- Soporte para gestión de grandes proyectos de desarrollo.
- Manual de PHP integrado.
- Soporte para control de versiones usando CVS o Subversion (a elección del desarrollador).
- Cliente FTP integrado.
- Soporte para navegación en bases de datos y ejecución de consultas del Lenguaje de Consultas Estructurado SQL (Structured Query Language).

NetBeans

El IDE NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de

sistemas para extender el IDE NetBeans. Es un producto libre y gratuito sin restricciones de uso.

NetBeans es un IDE de código abierto escrito completamente en Java usando la plataforma NetBeans. Además, soporta el desarrollo de todos los tipos de aplicación Java. Entre sus características se encuentra un sistema de proyectos basado en control de versiones. Todas las funciones del IDE son provistas por sistemas. Cada sistema provee una función bien definida, tales como el soporte de Java, edición, o soporte para el sistema de control de versiones.

NetBeans contiene todos los sistemas necesarios para el desarrollo de aplicaciones Java en una sola descarga, permitiéndole al usuario comenzar a trabajar inmediatamente. (Wheeler, 2008)

NetBeans IDE es un producto libre y gratuito sin restricciones de uso. Algunas de sus características son (Wheeler, 2008):

- Sintaxis resaltada.
- Completamiento de código y análisis de tecleo.
- Soluciones rápidas (Quick Fixes) y verificación de sintaxis.
- Refactorización.
- Generador de propiedades vean.
- Control de versión clearCase.
- Completamiento de código para parámetros, excepciones y otros.
- Plantillas de código.

Selección del IDE

Se ha seleccionado la herramienta NetBeans 7.4 como editor de PHP, pues ofrece un eficiente auto-completado de código, marcado de error de PHP, plantillas, macros, control de versiones, también es importante decir que es multiplataforma y de código abierto.

1.10 Servidor Web Apache

Es un programa que permite crear un servidor http en un ordenador de una forma rápida y sencilla. Está diseñado para ser un servidor web potente y flexible que pueda funcionar en las diferentes plataformas y entornos. Las diferentes plataformas y entornos, hacen que a menudo sean necesarias diferentes características o funcionalidades. Apache se

ha adaptado a los entornos a través de su diseño modular. Este diseño permite a los administradores de sitios web elegir qué características van a ser incluidas en el servidor seleccionando que sistemas se van a ejecutar, ya sea al compilar o al ejecutar el servidor. (Ciberaula.com, 2008)

Características principales (Ciberaula.com, 2008):

- Multiplataforma.
- Es un servidor web conforme al protocolo HTTP/1.1.
- Incentiva la realimentación de los usuarios, obteniendo nuevas ideas, informes de fallos y parches para la solución de los mismos.
- Se desarrolla de forma abierta.
- Extensible: gracias a ser modular se han desarrollado diversas extensiones entre las que se destaca PHP, un lenguaje de programación del lado del servidor.

1.11 Sistemas Gestores de Base de Datos

Un Sistema Gestor de Base Datos (SGBD) es el conjunto de programas que administran y gestionan la información contenida en una base de datos. Ayuda a llevar a cabo la definición de los datos, así como el mantenimiento de su integridad, el control de su seguridad, su privacidad y su manipulación. Un sistema gestor de base de datos está compuesto del gestor de la base de datos, que no es más que un conjunto de programas no visibles al usuario final que se encarga de la privacidad, integridad, la seguridad de los datos y la interacción con el sistema operativo. Proporcionando una interfaz entre los datos, los programas que los manejan y los usuarios finales. Entre los SGBD más utilizados en el mundo se encuentran Oracle, MySQL, Microsoft SQL Server, PostgreSQL, InterBase, entre otros. Todos estos presentan un enfoque relacional con un buen basamento matemático centrado en el Álgebra Relacional. Estos sistemas presentan disímiles ventajas, entre las que se encuentran (Alvarez, 2007):

- Facilidad de manejo de grandes volumen de información.
- Gran velocidad.
- Seguridad de la información (acceso a usuarios autorizados), protección de información, de modificaciones, inclusiones, consulta.

A continuación se muestran los SGBD estudiados.

Microsoft SQL Server

Es un SGBD relacional basado en el lenguaje SQL, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea. Además de brindar las ventajas que a continuación se pueden describir. (Microsoft, 2011)

Entre sus características figuran (Microsoft, 2011):

- Soporte de transacciones.
- Gran estabilidad.
- Gran seguridad.
- Escalabilidad.
- Soporta procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- Permite trabajar en modo cliente-servidor donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información.
- Además permite administrar información de otros servidores de datos.

Este sistema incluye una versión reducida, llamada Microsoft SQL Server Desktop Engine (MSDE) con el mismo motor de base de datos pero orientado a proyectos más pequeños. Microsoft SQL Server constituye la alternativa de Microsoft a otros SGBD como son Oracle o Sybase ASE. Es común desarrollar completos proyectos complementando Microsoft SQL Server y Microsoft Access a través de los llamados ADP (Access Data Project). De esta forma se completa una base de datos (Microsoft SQL Server) con un entorno de desarrollo cómodo y de alto rendimiento a través de la implementación de aplicaciones de dos capas mediante el uso de formularios Windows. Para el desarrollo de aplicaciones más complejas (tres o más capas), Microsoft SQL Server incluye interfaces de acceso para la mayoría de las plataformas de desarrollo, incluyendo .NET. Microsoft SQL Server, al contrario de MySQL, no es multiplataforma, ya que solo está disponible en Sistemas Operativos de Microsoft.

PostgreSQL

Es un servidor de base de datos relacional libre, es un motor de base de datos avanzado de código abierto. Es un sistema objeto-relacional, incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional.

Algunas de las características más importantes y soportadas por PostgreSQL (PostgreSQL.org, 2012):

- Es una base de datos 100% ACID (Atomicity, Consistency, Isolation and Durability) o lo que es lo mismo Atomicidad, Consistencia, Aislamiento y Durabilidad en español.
- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP), cadenas de bits, etc. También permite la creación de tipos propios.
- Incluye herencia entre tablas, por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Copias de seguridad en caliente (Online/hot backups)
- Unicode.
- Juegos de caracteres internacionales.
- Regionalización por columna.
- Multi-Version Concurrency Control (MVCC).
- Múltiples métodos de autentificación.
- Acceso encriptado vía SSL.
- Completa documentación.
- Licencia BSD.
- Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 32/64bit.

MySQL

Es un sistema de administración de bases de datos (Database Management System) para bases de datos relacionales. Es la base de datos de código fuente abierto más usada del mundo. Presenta condición de open source de MySQL, que hace que su utilización sea gratuita e incluso se pueda modificar con total libertad, pudiendo descargar su código fuente. Esto ha favorecido positivamente en su desarrollo y continuas actualizaciones, para hacer de MySQL una de las herramientas más utilizadas por los programadores orientados a Internet. Utiliza múltiples tablas para almacenar y organizar la información. Posee una adaptación a diferentes entornos de desarrollo, permitiendo su interactuación con los lenguajes de programación más utilizados como

PHP, Perl y Java y su integración en distintos sistemas operativos. Además muestra las siguientes ventajas (Packo, 2012):

- Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.
- Soporta gran cantidad de tipos de datos para las columnas.
- Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc).
- Gran portabilidad entre sistemas.
- Soporta hasta 32 índices por tabla.
- Gestión de usuarios y passwords, manteniendo un muy buen nivel de seguridad en los datos.
- Condición de open source de MySQL hace que la utilización sea gratuita y se puede modificar con total libertad.
- Se puede descargar su código fuente. Esto ha favorecido positivamente en su desarrollo y continuas actualizaciones.
- Gran rapidez y facilidad de uso.
- Librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación.
- Fácil instalación y configuración.

Selección del SGBD a utilizar

Después de haber analizado estos sistemas, se decide utilizar el servidor MySQL 5.3.1 ya que fue desarrollado originalmente para manejar grandes bases de datos mucho más rápido que las soluciones existentes, y está siendo usado en ambientes de producción sumamente exigentes por varios años. Aunque se encuentra en desarrollo constante, el servidor MySQL ofrece hoy un conjunto rico y útil de funciones. Su conectividad, velocidad, y seguridad hacen de MySQL un servidor apropiado para acceder a bases de datos en aplicaciones Web.

1.12 Conclusiones

Al término del capítulo y luego del análisis realizado se concluye que:

- El estudio de los diferentes HIS permitió conocer la carencia de sistemas que informaticen el proceso del cálculo de los índices pronósticos.

- Los sistemas estudiados se ajustan a las necesidades y características de las instituciones para la cuales fueron desarrollados, por lo que no son aplicables al campo de acción, además que no contemplan todo el proceso de la evolución del paciente (análisis diario del resultado de los índices pronósticos), el cálculo de los índices pronósticos y la gestión de las políticas de antibióticos.
- El estudio de las herramientas y tecnologías, permitió profundizar los conocimientos necesarios para el desarrollo de la propuesta solución, lo cual conlleva a utilizar a XP como metodología de desarrollo y Visual Paradigm 8.0 como herramienta de modelado. PHP 5.4.16 como lenguaje de programación, el marco de trabajo Symfony 2.3.7, el SGBD MySQL 5.3.1, el servidor web Apache 2.2.4 y el entorno de desarrollo NetBeans 7.4.

CAPITULO 2. PROPUESTA DE SOLUCIÓN

2.1 Introducción

En el capítulo anterior se determinó la poca existencia de herramientas que realicen el cálculo de los índices pronósticos, es por esto que se hace necesario desarrollar una sistema que gestione tanto la información de los pacientes, como la realización del cálculo de los índices pronósticos. En este capítulo se realiza todo el proceso de desarrollo aplicando la metodología seleccionada como respaldo al software a desarrollar. Se describen los aspectos no funcionales y los requisitos funcionales que deberá cumplir el sistema, las historias de usuario y la planificación del desarrollo de la herramienta.

2.2 Modelo conceptual

En el estudio de los procesos a automatizar se hizo un análisis de los principales conceptos, para modelar los mismos se utilizó un modelo conceptual. El objetivo principal del modelo conceptual es entender y detallar los conceptos más importantes dentro del contexto del sistema para capturar correctamente los requisitos y poder construir un proyecto que cumpla con las metas previstas.

Los principales conceptos del dominio son:

Paciente: persona que requiere asistencia sanitaria y está sometida a cuidados profesionales.

Enfermera: persona que se encarga de dar asistencia sanitaria a los pacientes.

Especialista: persona que se encarga de realizar los diagnósticos de los pacientes, así como también indicar las políticas de antibióticos.

Antibiótico: sustancia capaz de impedir el desarrollo o crecimiento de ciertos microorganismos, especialmente bacterias.

Política de antibióticos: antibióticos aplicados a un paciente determinado.

Índices pronósticos: permiten cuantificar y predecir el riesgo de morbilidad y mortalidad según parámetros fisiológicos, analíticos o clínicos, y su aplicación es una forma válida y rigurosa para medir la probabilidad de complicaciones y mortalidad postoperatoria.

Evolución del paciente: evolución diaria del paciente. Se clasifica en tres estados: crítico, grave y menos grave.

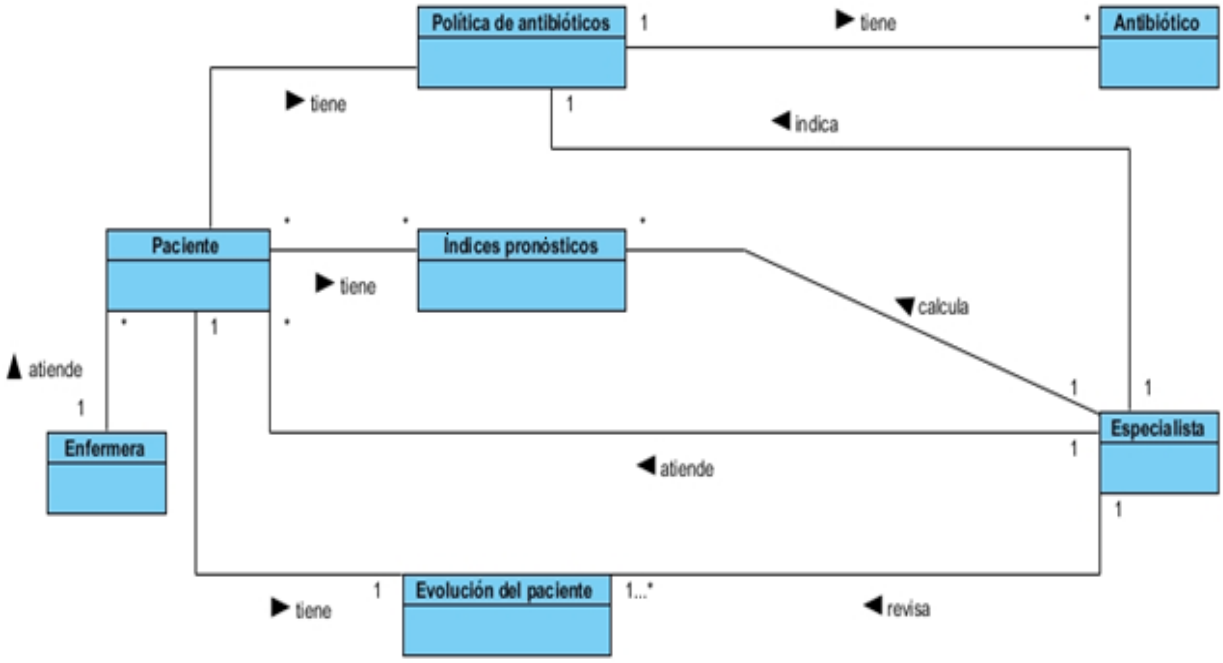

Figura 1 Modelo conceptual.

2.3 Exploración

La metodología de desarrollo XP comienza con su fase de Exploración, el cliente detalló las historias de usuario que el sistema debía poseer y que son de beneficio para la inicial entrega del producto. El equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. La fase de Exploración consta de poco tiempo según la familiaridad que tengan los programadores con la tecnología.

2.3.1 Usuarios del sistema

Se define como usuario del sistema a todo aquel que obtiene un resultado del valor de uno o varios procesos que se ejecutan en el mismo. También son los que se encuentran involucrados en dichos procesos, pues participan en ellos, pero no obtienen ningún resultado de valor.

Tabla 1 Usuarios del sistema.

Usuario	Descripción

Administrador	Puede realizar todas las funcionalidades definidas para el sistema, pudiendo gestionar, controlar y chequear la información de todos los usuarios registrados en el sistema.
Especialista	Encargado de introducir todos los datos de los pacientes, así como también realizar el cálculo de los índices pronósticos. Introducir la política de antibióticos y definir la evolución diaria del paciente.

2.3.2 Lista de reserva del producto

A través de la lista de reserva del producto se definen los aspectos funcionalidades y se describen los aspectos no funcionales que va a tener el sistema. Esta lista puede crecer y modificarse a medida que se obtiene más conocimiento acerca del producto y del cliente. El objetivo es asegurar que el producto definido al terminar la lista es el más correcto, útil y competitivo posible y para esto la lista debe acompañar los cambios en el entorno y el producto.

2.3.2.1 Aspectos no funcionales

Usabilidad

- Facilidad de uso por parte de los usuarios: el sistema debe presentar una interfaz que permita la fácil interacción con el mismo y llegar de manera rápida y efectiva a la información buscada. Debe, además, ser una interfaz de manejo cómodo que posibilite a los usuarios sin experiencia una rápida adaptación.
- Especificación de la terminología: el sistema debe adaptarse al lenguaje y términos utilizados por los clientes en la rama abordada, con vista a una mayor comprensión de la herramienta de trabajo.
- Emplear perfiles de usuario: diferenciar las interfaces y opciones para los usuarios que accedan al sistema según los diferentes roles que estos tengan dentro del mismo.
- Menús: el sistema debe presentar una serie de menús tanto laterales y superiores que permitan el acceso rápido a la información por parte de los usuarios, aprovechando así las potencialidades de estas estructuras.

Fiabilidad

- Seguridad de la base de datos: la base de datos deberá estar fraccionada en esquemas que permitan un mejor uso de la información. El SGBD escogido debe presentar facilidades de administración de roles y usuarios restringiendo el acceso a los datos.
- Políticas de seguridad por usuarios y roles: el sistema debe contar con un grupo de políticas de accesibilidad a las diferentes funcionalidades del mismo en dependencia del nivel de autorización que presente un usuario determinado.

Restricciones de diseño

- Lenguaje de programación: PHP 5.3 o superior. el marco de trabajo base de desarrollo que se utilizará es Symfony 2.3.7. Como IDE se empleará NetBeans 7.4. Como servidor web se explotará Apache 2.2.2. El SGBD deberá ser MySQL 5.3.1.

Interfaz

- Interfaz Web: la interfaz deberá ser ligera en cuanto a consumo de recursos y sin cúmulo de imágenes u objetos que distraigan al cliente del objetivo de su empleo.
- Interfaz interna: la interfaz interna estará determinada por los desarrolladores, construyendo así una vista escalable de las clases o agrupaciones de clases que permitirán un mejor encapsulamiento de las funcionalidades y una mayor abstracción modular del sistema.

Hardware

- Para explotación del cliente: PC Pentium 3 o superior, CPU 133 MHZ o superior, 256 RAM mínimo 512 RAM recomendada o superior.
- Para explotación del servidor: CPU Dual Core 2.0 GHZ o superior, memoria RAM de 2 GB (recomendado 4GB), 250 GB HDD.

2.3.2.2 Historias de Usuario

Las Historias de Usuario (HU) son la técnica utilizada para especificar los requisitos del software, sean funcionales o no funcionales, representar una breve descripción del comportamiento del sistema, donde se emplea la terminología del cliente sin lenguaje técnico.

Se realiza una HU por cada funcionalidad del sistema y cada una de estas deben ser lo suficientemente clara y determinada para que un programador pueda desarrollarla, se

emplean para hacer estimaciones de tiempo de desarrollo de la parte del sistema que se describe.

Las funcionalidades reflejadas en las HU deben ser programadas en un tiempo que lo define el propio equipo del proyecto que debe ser entre 1 y 3 semanas. Basado en las necesidades del proyecto y un estudio previo se propone a continuación las siguientes HU usando el modelo de la tabla 2 (Letelier, y otros, 2009):

Tabla 2 Plantilla de las historias de usuario.

Historia de usuario	
Número:	Usuario:
Nombre:	
Prioridad en el negocio:	Riesgo de desarrollo:
Puntos de estimación:	Iteración:
Programador responsable:	
Descripción:	

- Número: número de la HU (identificador).
- Usuario: usuario del sistema que realiza la acción (creador).
- Nombre: nombre de la HU.
- Prioridad en el negocio: se refiere a la importancia que tiene la HU para el cliente o el equipo de desarrollo la cual se define en la escala de media, alta o baja.
- Riesgo en desarrollo: nivel de dificultad para desarrollar la HU por el desarrollador.
- Puntos de Estimación: es el tiempo estimado para realizar la actividad la cual la establecen los programadores y varían en dependencia de la complejidad de la HU.
- Iteración: iteración a la que pertenece.
- Programador Responsable: el encargado de programar la actividad.
- Descripción: se describe en qué consiste la HU teniendo en cuenta las acciones de los usuarios y la respuesta del sistema.

Historias de usuario a desarrollar:

- Adicionar paciente.

- Modificar paciente.
- Listar paciente.
- Adicionar política de antibióticos.
- Modificar política de antibióticos.
- Eliminar política de antibióticos.
- Listar política de antibióticos.
- Adicionar antibióticos.
- Modificar antibióticos.
- Eliminar antibióticos.
- Listar antibióticos.
- Adicionar evolución del paciente.
- Eliminar evolución del paciente.
- Calcular índices pronósticos.
- Adicionar índices pronósticos.
- Modificar índices pronósticos.
- Eliminar índices pronósticos.
- Listar índices pronósticos.
- Realizar búsqueda de pacientes.
- Listar historial.

A continuación se muestran una parte de las HU, el resto se encuentran en el anexo 1:

Tabla 3 HU Adicionar paciente.

Historia de usuario	
Número: 1	Usuario: Usuario
Nombre: Adicionar paciente	
Prioridad en el negocio: alta	Riesgo de desarrollo: medio
Puntos de estimación: 1	Iteración: 1
Programador responsable: Lázaro E. Millares Bermúdez	
Descripción: permite registrar pacientes en el sistema, adicionando todos los datos obligatorios (número de historia clínica, sexo, raza, nombre, apellidos, dirección, carnet de identidad) y campos opcionales (interrogatorios por aparatos, antecedentes patológicos, examen físico, diagnósticos y operaciones) en dependencia de las características propias del paciente. Luego de añadido el paciente se mostrará una tabla con los datos insertados.	

Tabla 4 HU Adicionar política de antibióticos.

Historia de usuario	
Número: 2	Usuario: Usuario
Nombre: Adicionar política de antibióticos	
Prioridad en el negocio: alta	Riesgo de desarrollo: medio
Puntos de estimación: 1	Iteración: 1
Programador responsable: Lázaro E. Millares Bermúdez	
Descripción: el usuario tendrá la posibilidad de añadir políticas de antibióticos (antibiótico indicado, dosis, escala (miligramos, gramos, unidades)) a cada paciente que se encuentre registrado en el sistema. Después de insertada se debe mostrar un diagrama de Gantt con todas las políticas de antibióticos indicadas (activas o no) y una lista de las mismas.	

Tabla 5 HU Adicionar evolución del paciente.

Historia de usuario	
Número: 3	Usuario: Usuario
Nombre: Adicionar evolución del paciente	
Prioridad en el negocio: alta	Riesgo de desarrollo: media
Puntos de estimación: 1	Iteración: 1
Programador responsable: Frank Delfín Reyes	
Descripción: el usuario podrá añadir un estado de evolución (menos grave, grave, crítico) diario del paciente, luego de añadido el mismo se debe mostrar un gráfico con los porcientos de cada categoría de evolución del paciente.	

Tabla 6 HU Calcular índices pronósticos.

Historia de usuario	
Número: 4	Usuario: Usuario
Nombre: Calcular índices pronósticos.	
Prioridad en el negocio: alta	Riesgo de desarrollo: media
Puntos de estimación: 2	Iteración: 1
Programador responsable: Frank Delfín Reyes	
Descripción: el sistema debe permitir el cálculo de los índices pronósticos (APACHE II, IPM, SS, FMO, Possum, SSC) mediante la inserción de las variables. Después de insertadas estas, el sistema mostrará el valor calculado en una gráfica de línea para el seguimiento de los mismos, y una lista con todos los predictores graficados.	

2.4 Planeación

En esta fase el cliente define la prioridad que posee cada HU, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Las estimaciones de esfuerzo asociado a la implementación de las HU son establecidas por los programadores utilizando como medida la cantidad de puntos, por lo que un punto equivale a una semana de programación y generalmente las HU valen de 1 a 3 puntos. Se toman acuerdos sobre el contenido y plazo de la primera entrega modelándolo en un cronograma, estas actividades son realizadas en conjunto con el cliente. Las planificaciones serán respetadas por todos los implicados en el desarrollo y permitirán un desarrollo ágil y eficiente del sistema. (Letelier, y otros, 2009)

2.4.1 Estimación de esfuerzo por HU

En la siguiente tabla se muestran los resultados de la estimación realizada con cada una de las HU determinadas para la elaboración del sistema propuesto.

Tabla 7 Estimación de esfuerzo por HU.

Historia de usuario	Puntos de estimación
Adicionar paciente	1
Modificar paciente	1
Listar paciente	1
Adicionar política de antibióticos	2
Modificar política de antibióticos	2
Eliminar política de antibióticos	1
Listar política de antibióticos	1
Adicionar antibióticos	1
Modificar antibióticos	1
Eliminar antibióticos	1
Listar antibióticos	1
Adicionar evolución del paciente	2
Eliminar evolución del paciente	2
Calcular índices pronósticos	1
Adicionar índices pronósticos	1
Modificar índices pronósticos	1
Eliminar índices pronósticos	1
Listar índices pronósticos	1
Realizar búsqueda de pacientes	1
Listar historial	1

2.4.2 Planificación de iteraciones

Después de haber identificado las HU por el cliente y la estimación de esfuerzo dedicado a cada una de ellas por los desarrolladores, se procede a realizar tantas iteraciones como se consideren necesarias, definiendo cuales serán desarrolladas en cada iteración del proceso de implementación. Se decide implementar el sistema en 2 iteraciones:

Primera Iteración:

Para esta iteración se seleccionaron las HU básicas del sistema y las que permiten modelar la estructura del sistema. Al concluir dicha iteración se contará con una nueva versión del producto (1.0), las HU a implementar en esta iteración son:

- Adicionar paciente.
- Modificar paciente.
- Listar paciente.
- Calcular índices pronósticos.
- Adicionar índices pronósticos.
- Modificar índices pronósticos.
- Eliminar índices pronósticos.
- Listar índices pronósticos.
- Adicionar evolución del paciente.
- Eliminar evolución del paciente.

Segunda iteración:

En esta última iteración se seleccionaron las HU restantes. Al concluir esta iteración se obtendrá el producto en su versión (1.1), además el sistema podrá ser sometido a diversos procesos de prueba para comprobar su eficiencia y desempeño, las HU para esta iteración son:

- Adicionar política de antibióticos.
- Modificar política de antibióticos.
- Eliminar política de antibióticos.
- Listar política de antibióticos.
- Adicionar antibióticos.
- Modificar antibióticos.
- Eliminar antibióticos.
- Listar antibióticos.
- Realizar búsqueda de pacientes.
- Listar historial.

2.4.3 Plan de duración de iteraciones

Se propone la realización de un plan de duración de iteraciones con el fin de ilustrar la planificación temporal de las diferentes iteraciones y las HU que serán implementadas en cada una de ellas.

Tabla 8 Plan de duración de iteraciones.

Iteración	Historia de usuario	Duración
1	Adicionar paciente	11 semanas
	Modificar paciente	
	Listar paciente	
	Calcular índices pronósticos	
	Adicionar índices pronósticos	
	Modificar índices pronósticos	
	Eliminar índices pronósticos	
	Listar índices pronósticos	
	Adicionar evolución del paciente	
	Eliminar evolución del paciente	
2	Adicionar política de antibióticos	12 semanas
	Modificar política de antibióticos	
	Eliminar política de antibióticos	
	Listar política de antibióticos	
	Adicionar antibióticos	
	Modificar antibióticos	
	Eliminar antibióticos	
	Listar antibióticos	
	Realizar búsqueda de pacientes	
	Listar historial	

2.4.4 Plan de entregas

A continuación se muestra el plan de entregas que es una planificación donde los desarrolladores y los clientes establecen los períodos de implementación a través de las HU, clasificando las mismas según sus prioridades. Se establece efectuar dos entregas, estas son:

Tabla 9 Plan de entregas.

Iteración	Duración de la iteración	Fecha inicio	Fecha fin
1	11 semanas	1/15/2014	4/7/2014
2	12 semanas	4/20/2014	7/19/2014

2.5 Diseño del sistema

La metodología XP propone técnicas como las tarjetas CRC (Clases, Responsabilidad y Colaboración) la cual permite al programador centrarse y apreciar el desarrollo. Una clase es cualquier persona, objeto, evento, concepto, pantalla o reporte. Las responsabilidades de una clase son los elementos que conoce y las acciones que realiza sobre ellos, sus atributos y métodos. Los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades. (Letelier, y otros, 2009)

2.5.1 Tarjetas C.R.C

Las tarjetas C.R.C reducen al mínimo la complejidad del diseño. Esto logra que el programador se centre en el diseño de las bases fundamentales de la clase. Permiten que el equipo completo contribuya en la tarea del diseño. Se utilizan para estructurar las clases y definir las responsabilidades sobre las mismas, así como la simulación de escenarios en el sistema. Se propone la utilización del modelo de la tabla 10 (Letelier, y otros, 2009):

Tabla 10 Plantilla de las Tarjetas CRC.

Tarjeta CRC	
Clase*:* nombre de la clase que se está modelando	
Responsabilidades: es una descripción del propósito de la clase	Colaboraciones: indica con cuáles otras clases se requiere relación para cumplir la responsabilidad

A continuación se presentan las tarjetas CRC correspondientes a la propuesta de solución:

Tabla 11 CRC pacienteController.

Tarjeta CRC
Clase: pacienteController

Responsabilidades: es la clase encargada de gestionar todo lo referente a los pacientes.	**Colaboraciones:** paciente

Tabla 12 CRC antibioticoController.

Tarjeta CRC	
Clase: antibioticoController	
Responsabilidades: es la clase encargada de gestionar todo lo referente a los antibióticos.	**Colaboraciones:** antibiótico

Tabla 13 CRC evolucionController.

Tarjeta CRC	
Clase: evolucionController	
Responsabilidades: es la clase encargada de gestionar todo lo referente a la evolución.	**Colaboraciones:** paciente

Tabla 14 CRC politicaAntibioticosController.

Tarjeta CRC	
Clase: politicaAntibiticosController	
Responsabilidades: es la clase encargada de gestionar todo lo referente a las políticas de antibióticos usadas.	**Colaboraciones:** paciente antibiotíco

Tabla 15 CRC apacheIIController.

Tarjeta CRC
Clase: apacheIIController

Responsabilidades: es la clase encargada de gestionar todo lo referente al predictor APACHEII.	**Colaboraciones:** paciente

Tabla 16 CRC arpiController.

Tarjeta CRC	
Clase: arpiController	
Responsabilidades: es la clase encargada de gestionar todo lo referente al predictor ARPI.	**Colaboraciones:** paciente

2.5.2 Patrón arquitectónico

Symfony está basado en el patrón de diseño web conocido como arquitectura Modelo-Vista-Controlador (MVC), el cual es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo, es el sistema de gestión de base de datos y la lógica de negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista, teniendo como finalidad mejorar la reusabilidad por medio del desacople entre la vista y el modelo. (Sebastián, 2010)

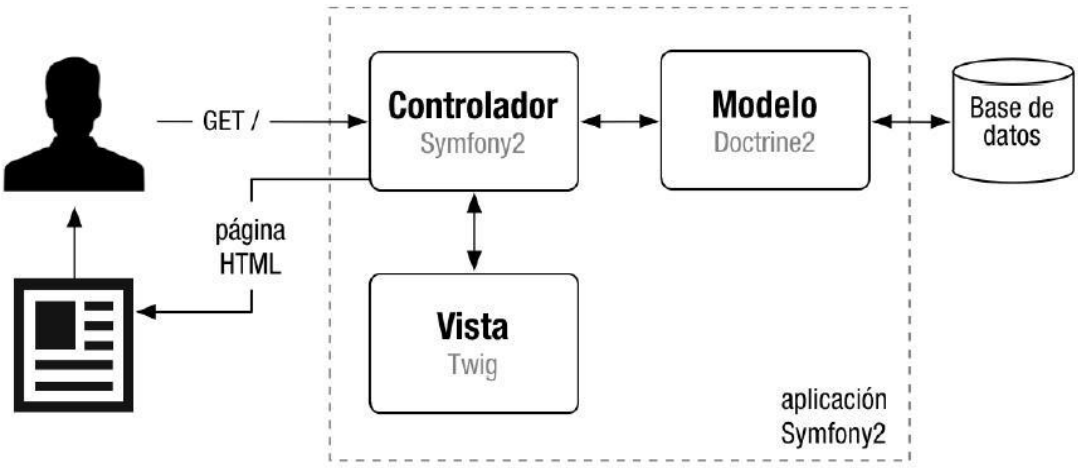

Figura 2 Modelo Vista Controlador

A continuación se muestran las responsabilidades de las capas basadas en la arquitectura MVC (Sebastián, 2010):

El modelo es el responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Definir las reglas de negocio.

El controlador es el responsable de:

- Recibir los eventos de entrada.
- Contiene reglas de gestión de eventos. Estas acciones pueden suponer peticiones al modelo o a las vistas.

Las vistas son responsables de:

- Recibir datos del controlador y mostrarlos al usuario.
- Tener un registro de su controlador asociado.
- Pueden dar el servicio de “Actualización”, para que sea invocado por el controlador.

Dentro de las ventajas que este patrón de diseño web posee se encuentran (Sebastián, 2010):

- La posibilidad de tener diferentes vistas para un mismo modelo.
- La construcción de nuevas vistas sin necesidad de modificar el modelo subyacente.
- Proporciona un mecanismo de configuración a componentes complejos mucho más tratables que el puramente basado en eventos, ya que el modelo puede verse como una representación estructurada del estado de la interacción.

2.5.3 Diseño de la base de datos

El correcto funcionamiento de la aplicación depende también del buen diseño y funcionamiento de la base de datos a utilizar. En este epígrafe se muestran el Diagrama de Entidad Relación (DER), correspondiente al diseño de la BD.

El DER representa la realidad de la problemática identificada a través de las entidades y de los enlaces que rigen la unión de las mismas y que constituyen a relación del modelo. Es el modelo conceptual más utilizado en el diseño de BD, capaz de mostrar cómo funcionará el sistema y la información que almacenará. Los DER son una

necesidad para diseñar y mantener una buena BD relacional, son una forma gráfica de representarla. (Alvarez, 2007)

Para el presente sistema se representaron un total de 13 tablas, interrelacionadas unas con otras. Para el diseño del DER se estableció un estándar para organizar el trabajo, representando los nombres de las tablas y sus atributos en mayúsculas y en el caso de los nombres compuestos, se diferencian con mayúsculas.

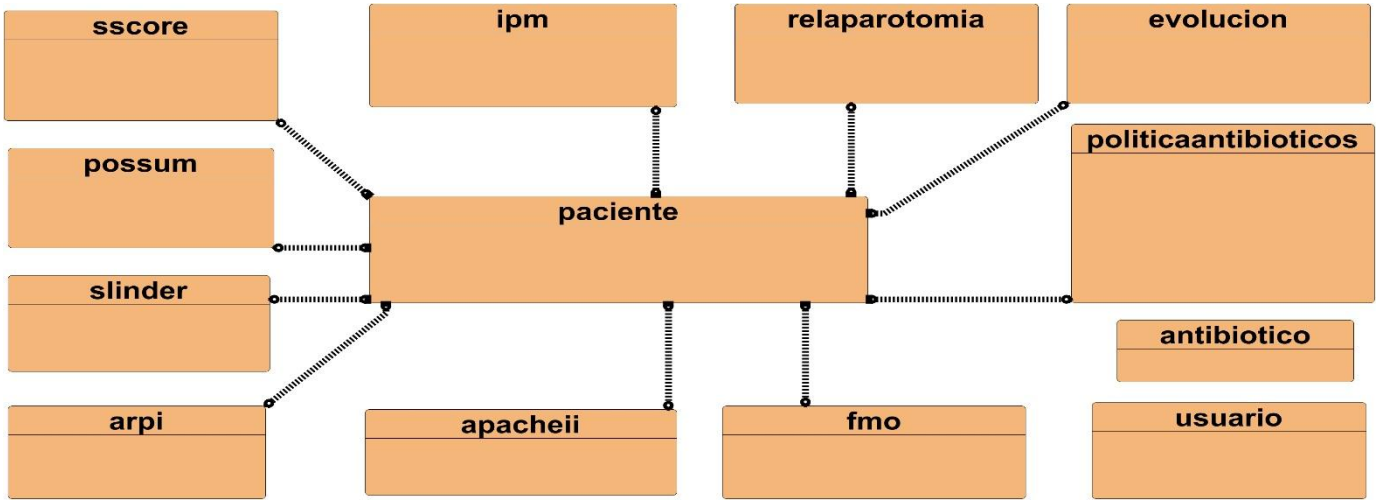

Figura 3 Diagrama entidad relación

Descripción de las tablas:

paciente: es la entidad encargada de recoger los datos de un paciente en el sistema.

Índices pronósticos (sscore, ipm, possum, slinder, arpi, apacheii, fmo)**:** son las entidades encargadas de recoger los datos de las variables pronóstico.

relaparotomia: entidad encargada de recoger los datos de las relaparotomías de los pacientes.

evolucion: entidad encargada de recoger los datos de la evolución de los pacientes.

politicaantibioticos: entidad encargada de recoger los datos de las políticas de antibióticos de los pacientes.

antibiotico: entidad encargada de recoger los nombres de los antibióticos disponibles en el sistema.

usuario: entidad encargada de recoger los datos de los usuarios registrados en el sistema.

2.6 Conclusiones

Durante la elaboración de este capítulo, el estudio de las fases de Exploración y Planificación propias de la metodología de desarrollo utilizada para la implementación del sistema, permite arribar a las siguientes conclusiones:

- Durante la Fase de Exploración se describen las HU en correspondencia con cada uno de los aspectos identificados y se hace una descripción de todos los artefactos generados mediante las mismas, para realizar una implementación con el mínimo de esfuerzo por parte de los programadores.
- Durante la Fase de Planificación se realiza la estimación del esfuerzo por cada HU; se realiza el plan de iteraciones y el plan de duración de cada iteración y se define un plan de entregas con las propuestas de liberación de las versiones del sistema, que permiten tener un estimado de tiempo para el desarrollo.

CAPÍTULO 3 VALIDACIÓN DE LA SOLUCIÓN

3.1 Introducción

En presente capitulo se detallan las iteraciones de implementación llevadas a cabo durante la etapa de construcción del sistema, exponiéndose las tareas de programación o ingeniería generadas por cada historia de usuario. Además, se realizan las pruebas correspondientes, donde solo se podrá liberar una nueva versión, si esta ha superado el cien por ciento de la totalidad de ellas. Se analiza el resultado de las pruebas con el objetivo de lograr un sistema completamente funcional y con la calidad requerida por el cliente.

3.2 Implementación

En la fase de Planificación se detallaron las HU agrupadas en cada iteración. En esta fase se descomponen las HU en tareas de ingeniería (TI). Las tareas de ingeniería se efectúan para facilitar con ellas el entendimiento en el proceso de implementación. Cada HU puede poseer una o más tareas de ingeniería en caso de ser necesario, explicando paso a paso las acciones que se realizan para dar cumplimiento a la misma. A partir de la planificación efectuada se llevaron a cabo dos iteraciones de desarrollo, lo que permite al final de la última iteración obtener un sistema que cumple con todas las funcionalidades definidas. A continuación se detallan cada una de las iteraciones efectuadas.

3.2.1 Primera Iteración

En esta iteración se desarrollaron las funcionalidades básicas del sistema que darán soporte a las funciones más complejas de la segunda iteración.

Tabla 17 HU desarrolladas en la primera iteración.

Historia de Usuario	Estimado	Real
Adicionar paciente	1	1
Modificar paciente	1	1
Listar paciente	1	1
Calcular índices pronósticos	1	1
Adicionar índices pronósticos	1	1
Modificar índices pronósticos	1	1
Eliminar índices pronósticos	1	1
Listar índices pronósticos	1	1

Adicionar evolución del paciente	2	2
Eliminar evolución del paciente	2	2

A continuación se muestran las tareas de ingeniería correspondiente a las HU (Adicionar paciente, Calcular índices pronósticos, Adicionar evolución del paciente), el resto se pueden consultar en el anexo 2:

Tabla 18 TI Adicionar paciente.

Tarea de ingeniería	
No. De la tarea:1	No. De la HU:1
Nombre de la terea: Adicionar paciente	
Tipo de tarea: desarrollo	Puntos estimados: 0.1
Fecha inicio:1/15/2014	Fecha fin:1/22/2014
Programador responsable: Lázaro E. Millares Bermúdez	
Descripción: implementar una interfaz que permita la entrada de los datos necesarios para añadir un paciente Los datos son: Nombre, número de carnet de identidad, número de historia clínica, apellidos, edad, dirección.	

Tabla 19 TI Calcular índices pronósticos.

Tarea de ingeniería	
No. De la tarea:2	No. De la HU:4
Nombre de la terea: Calcular índices pronósticos	
Tipo de tarea: desarrollo	Puntos estimados: 0.3
Fecha inicio:1/22/2014	Fecha fin:1/29/2014
Programador responsable: Frank Delfín Reyes	
Descripción: desarrollar las funcionalidades necesarias que permitan el cálculo de los índices pronósticos.	

Tabla 20 TI Adicionar evolución del paciente.

Tarea de ingeniería	
No. De la tarea:3	No. De la HU:9
Nombre de la terea: Adicionar evolución del paciente	
Tipo de tarea: desarrollo	Puntos estimados: 0.5

Fecha inicio:1/30/2014	Fecha fin:2/5/2014
Programador responsable: Lázaro E. Millares Bermúdez	
Descripción: implementar una interfaz que permita definir un estado de gravedad de un paciente. Los estados de gravedad son: menos graves, graves, críticos.	

3.2.2 Segunda iteración

En esta iteración se desarrollaron el resto de las funcionalidades para culminar el desarrollo del sistema.

Tabla 21 HU Desarrolladas en la segunda iteración.

Historia de Usuario	Estimado	Real
Adicionar política de antibióticos	2	2
Modificar política de antibióticos	2	2
Eliminar política de antibióticos	1	1
Listar política de antibióticos	1	1
Adicionar antibióticos	1	1
Modificar antibióticos	1	1
Eliminar antibióticos	1	1
Listar antibióticos	1	1
Realizar búsqueda de pacientes	1	1
Listar historial	1	1

A continuación se muestran una parte de las tareas de ingeniería correspondiente a las HU (Adicionar política de antibióticos, Adicionar antibióticos), el resto se pueden consultar en el anexo 2:

Tabla 22 TI Adicionar política de antibióticos.

Tarea de ingeniería	
No. De la tarea:4	No. De la HU:11
Nombre de la terea: Adicionar política de antibióticos	
Tipo de tarea: desarrollo	Puntos estimados: 0.5
Fecha inicio:2/6/2014	Fecha fin:2/15/2014
Programador responsable: Lázaro E. Millares Bermúdez	

Descripción: implementar una interfaz que permita la entrada de los datos necesarios para añadir una política de antibióticos. Los datos son: Dosis, antibiótico.

Tabla 23 TI Adicionar antibióticos.

Tarea de ingeniería	
No. De la tarea:5	No. De la HU:15
Nombre de la terea: Adicionar antibióticos	
Tipo de tarea: desarrollo	Puntos estimados: 0.3
Fecha inicio:2/16/2014	Fecha fin:2/22/2014
Programador responsable: Frank Delfín Reyes	
Descripción: implementar una interfaz que permita la entrada de los datos necesarios para añadir un antibiótico. Los datos son: Nombre.	

3.3 Análisis del funcionamiento del sistema

El sistema está compuesto por 4 módulos (Paciente, Antibióticos, Resumen de UCI (Unidad de Cuidados Intensivos), Historial pacientes) cumpliendo con la propuesta de los procesos automatizables definidos en el Capítulo 2 e implementados en el epígrafe anterior. Permite además, la entrada de los datos para el cálculo de los índices pronósticos, visualizar la información de un paciente determinado, así como también analizar su evolución. A continuación se explica el funcionamiento de los módulos mencionados anteriormente.

3.3.1 Módulo Paciente

Gestiona toda la información de los pacientes cuando ingresan en el sistema, además muestra un listado de los pacientes ingresados en el sistema y permite el cálculo de los índices pronósticos. Controla la evolución diaria de un paciente determinado y su política de antibióticos. A continuación se muestran las vistas para dichas funcionalidades, figuras 4, 5, 6 y 7.

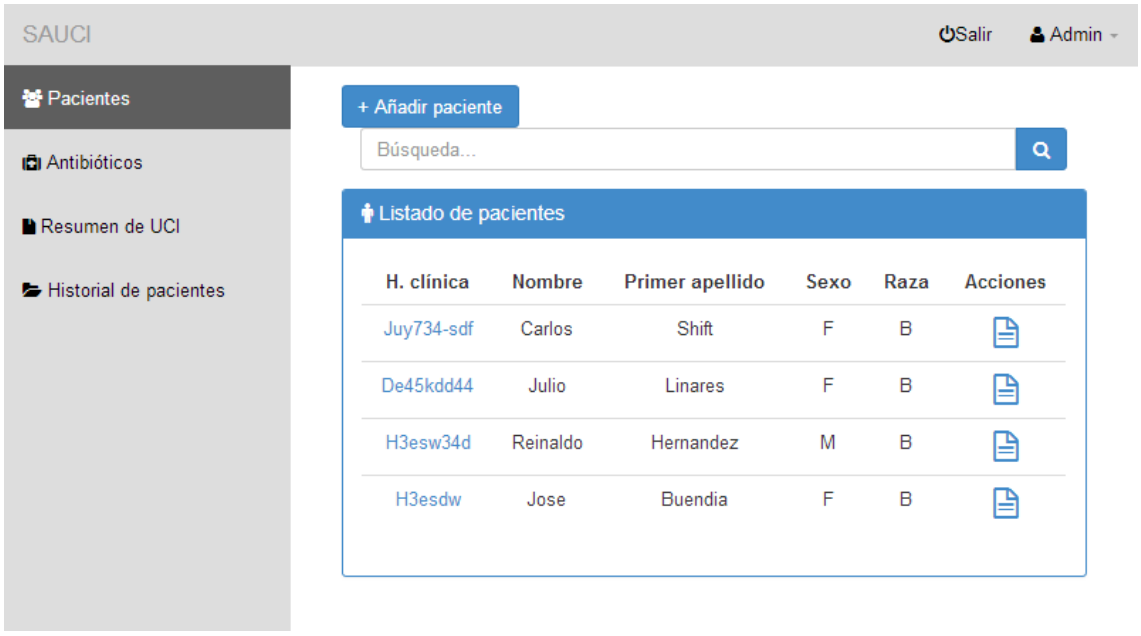

Figura 4 Listado de pacientes.

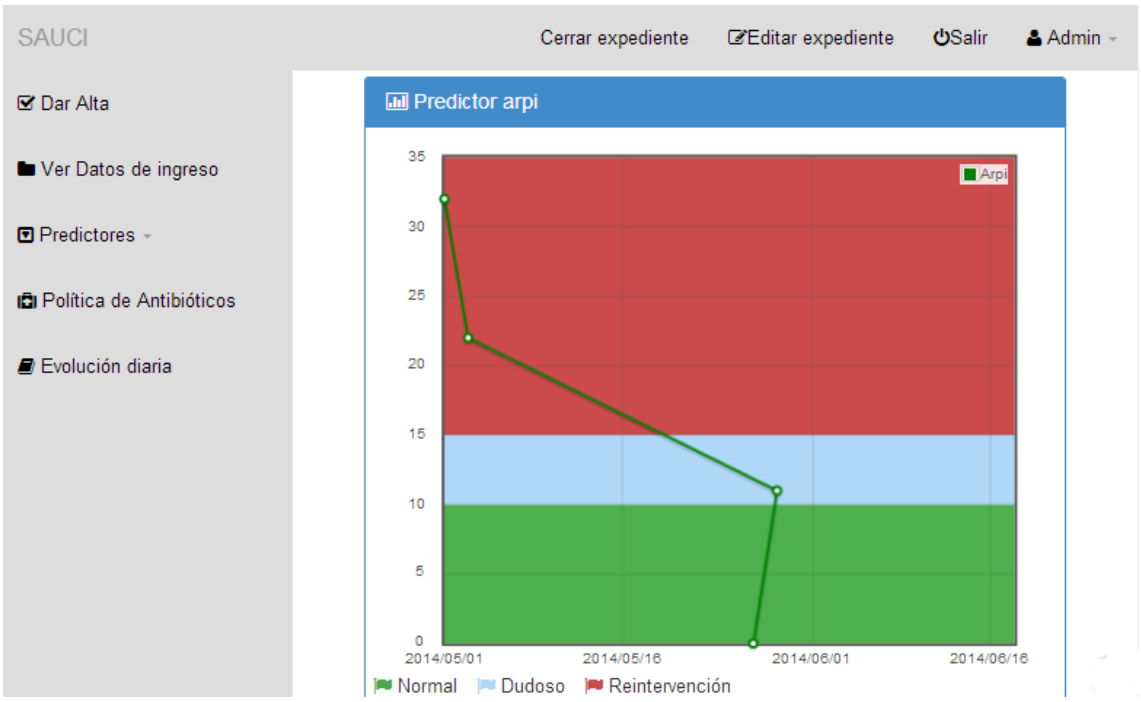

Figura 5 Índice Pronósticos.

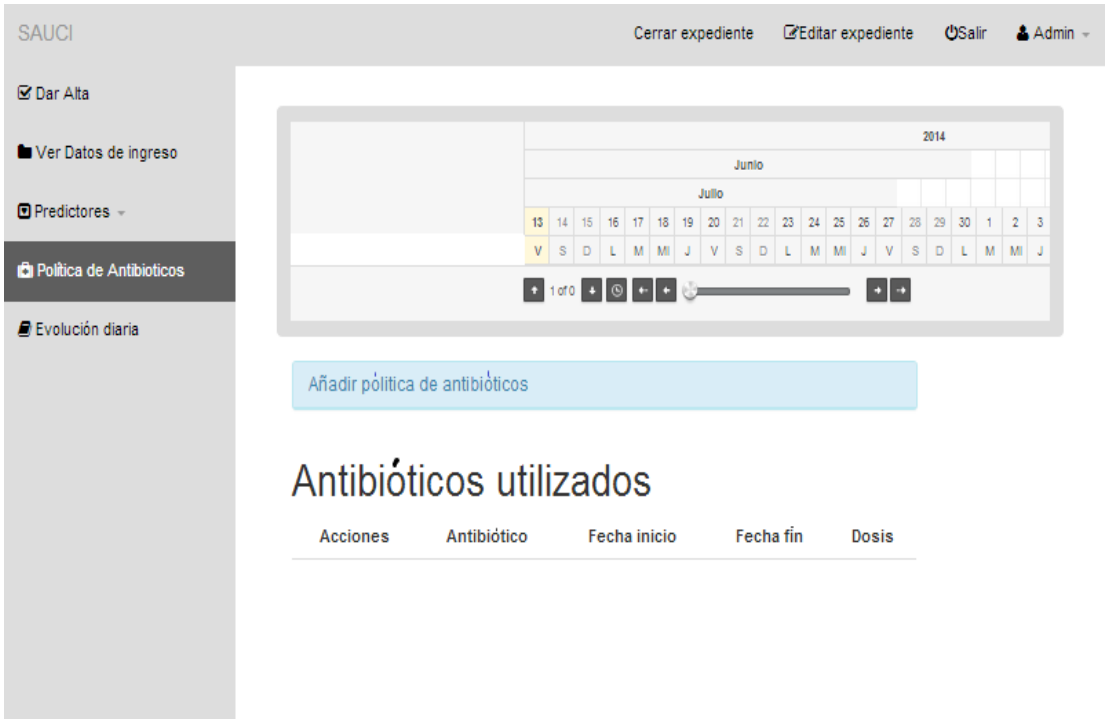

Figura 6 Política de antibióticos.

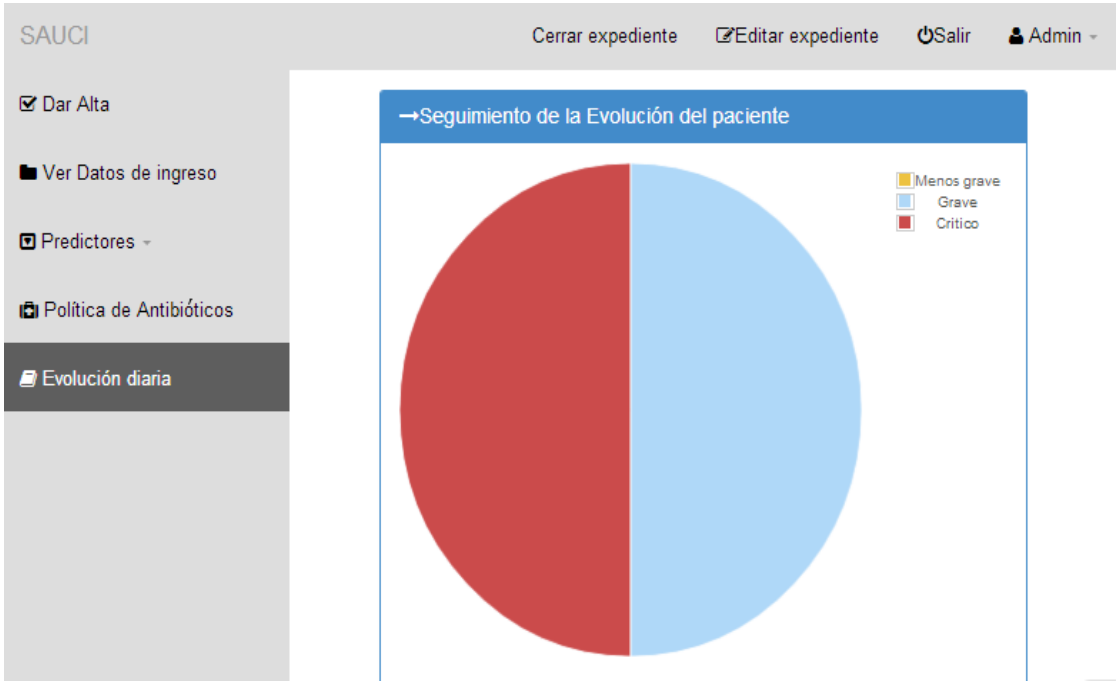

Figura 7 Evolución diaria.

3.3.2 Módulo Antibióticos

Gestiona los antibióticos existentes en la unidad de cuidados intensivos (UCI), permitiéndole a los especialistas disponer de estos para aplicar las políticas de antibióticos a los pacientes ingresados. A continuación se muestra la vista de esta funcionalidad, figura 8.

Figura 8 Módulo Antibióticos.

3.3.3 Módulo Resumen de UCI

Resumen diario de la actividad (Operados, Estado de gravedad, Cantidad de sangre transfundida, Edad, Sexo, Egresos) en la unidad de cuidados intensivos, permitiendo realizar un control sobre el estado en general de la unidad de cuidados intensivos, para la toma de decisiones en caso de eventos especiales (Evacuación). Se presenta una vista de la funcionalidad en la figura 9.

Figura 9 Resumen de UCI.

3.3.4 Módulo Historial pacientes

Lista todos los pacientes dados de alta. En el caso de los pacientes dados de alta (vivos), pueden ser reingresados. Esto permite acelerar el proceso de ingreso puesto que los datos del mismo ya están en el sistema. Esta funcionalidad se muestra en la figura 10.

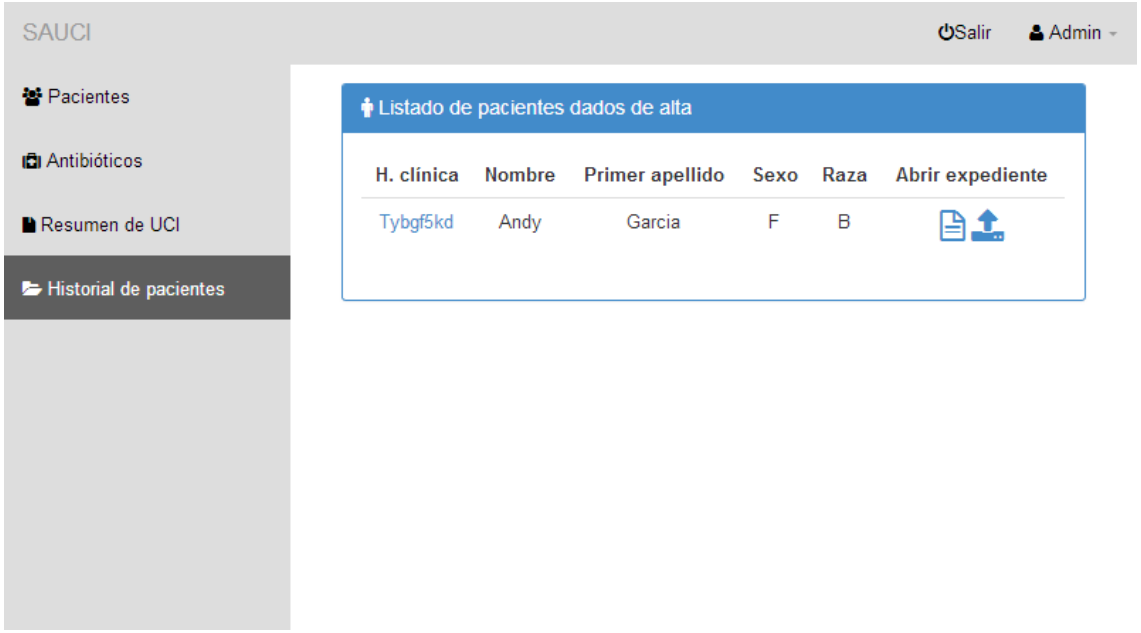

Figura 10 Historial pacientes.

3.4 Pruebas

Uno de los principios fundamentales de XP es el proceso de prueba, donde los desarrolladores realizan pruebas constantemente, tanto como sea posible; de esta manera se reduce el número de errores no detectados, así como el tiempo entre la introducción de éste en el sistema y su detección. Todo esto contribuye a elevar la calidad del producto desarrollado.

XP divide las pruebas en dos grupos (Gutierrez, y otros, 2010):

- Pruebas unitarias, desarrolladas por los programadores, encargadas de verificar el código de forma automática.
- Pruebas de aceptación, las cuales son destinadas a evaluar bloques más grandes de la funcionalidad del sistema, tales como las HU, verificando que al final de una iteración se obtuvo la funcionalidad requerida, además de que dicha funcionalidad sea la esperada por el cliente.

3.4.1 Pruebas unitarias

De acuerdo con lo que plantea la metodología XP, las pruebas unitarias o pruebas de unidad consisten en comprobaciones (manuales o automatizadas) desarrolladas por los programadores. Las cuales se realizan para verificar que el código correspondiente a un módulo concreto se comporta de manera esperada. Las pruebas unitarias proporcionan beneficios tales como (Gutierrez, y otros, 2010):

- Brindan al programador una inmediata retroalimentación de cómo está realizando su trabajo.
- El programador puede realizar cambios de forma segura respaldada por efectivos casos de prueba.
- Permite saber si una determinada funcionalidad se puede agregar al sistema existente sin alterar el funcionamiento actual del mismo.

Se realizaron pruebas unitarias a las entidades más críticas (POSSUM, APACHE II, ARPI, SLINDER, SSCORE, IPM) usando la herramienta PHPUnit, con el objetivo verificar que los datos que se almacenaran sea los indicados.

PHPUnit es un framework que facilita la creación de clases de pruebas sobre aplicaciones basadas en PHP. (Merayo Castellano, 2012)

Entre las características de PHPUnit, se encuentran (Merayo Castellano, 2012):

- Un gran API de aserciones.
- Una línea de comando, que nos permite exportar a diferentes formatos los resultados (JUnit, TAP, JSON), generar informes de cobertura de código (Clover XML, HTML o texto), entre otros.
- Database testing.
- Desarrollo guiado por comportamiento (Behavior-Driven Development).

Estas pruebas fueron realizadas al finalizar cada iteración de la fase de implementación. A continuación se muestra el resultado de las mismas en la figura 11:

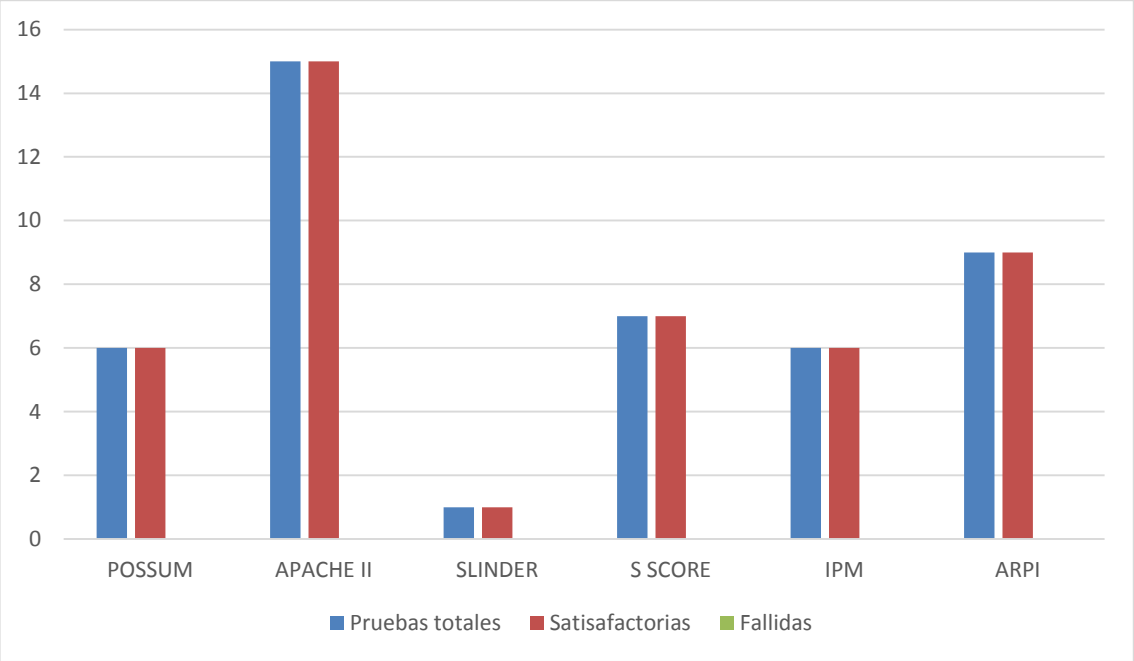

Figura 11 Resultado de las pruebas unitarias.

3.4.2 Pruebas de aceptación

Las pruebas de aceptación significan la satisfacción del cliente con el producto desarrollado y el final de una iteración y el comienzo de la siguiente. Las mismas se realizan a lo largo de la iteración, en paralelo con el desarrollo del sistema y adaptándose a los cambios que sufra el mismo, son consideradas como “pruebas de caja negra”.

Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. Así mismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una HU no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación. (Gutierrez, y otros, 2010)

Tabla 24 Prueba de aceptación.

Pruebas de aceptación	
Código:	No. HU
Nombre:	
Descripción:	
Condiciones de ejecución:	
Entrada:	
Resultado esperado:	
Evaluación de la prueba:	

- Código: índice del caso de prueba. Es un número único que se le asigna a cada caso de prueba que pertenece a una HU determinada con el fin de lograr una mejor organización de estos.
- Nombre: nombre del caso de prueba. Debe ser descriptivo, en la medida de las posibilidades, de lo que se comprobará y no muy extenso.
- No. Historia de Usuario: número de la HU a la que corresponde. Índice de la historia de usuario a la que se le desea comprobar este aspecto.
- Descripción: se describe qué es lo que se desea probar. La descripción debe ser corta y precisa.
- Condiciones de Ejecución: condiciones especiales que deben tenerse en cuenta para ejecutar el caso de prueba.
- Entrada: entrada al caso de prueba en caso de necesitarlas.
- Resultado Esperado: resultado que se desea que tenga el caso de prueba. Descripción breve de lo que debe suceder.
- Evaluación de la Prueba: se evalúa si el caso de prueba tuvo éxito o no. En caso de ser exitoso se asigna un resultado satisfactorio, en caso contrario no satisfactorio.

A continuación se muestran las pruebas de aceptación de las siguientes HU (Adicionar paciente, Adicionar política de antibióticos, Adicionar evolución del paciente, Calcular índices pronósticos), se pueden consultar el resto en el anexo 3:

Tabla 25 PA Adicionar paciente.

Pruebas de aceptación	
Código: 1	No. HU: 1
Nombre: Adicionar paciente	
Descripción: prueba para crear un nuevo paciente en el sistema.	
Condiciones de ejecución: que el usuario esté autentificado en el sistema	
Entrada: se realiza la entrada de los datos válidos y no duplicados en el sistema. Datos: nombre, edad, sexo, entre otros.	
Resultado esperado: se crea un nuevo paciente en el sistema con los datos correspondientes.	
Evaluación de la prueba: prueba satisfactoria.	

Tabla 26 PA Adicionar política de antibióticos.

Pruebas de aceptación

Código: 2	No. HU: 2
Nombre: Adicionar política de antibióticos	
Descripción: prueba para adicionar una política de antibiótico a un paciente determinado.	
Condiciones de ejecución: que el usuario esté autentificado en el sistema, que exista el paciente, y exista un antibiótico.	
Entrada: se realiza la entrada de los datos válidos. Datos: antibiótico, dosis diaria, unidad de medida.	
Resultado esperado: se adiciona una política de antibióticos a un paciente.	
Evaluación de la prueba: prueba satisfactoria.	

Tabla 27 PA Adicionar evolución del paciente.

Pruebas de aceptación	
Código: 3	No. HU: 3
Nombre: Adicionar evolución del paciente.	
Descripción: prueba para adicionar una evolución diaria a un paciente.	
Condiciones de ejecución: que el usuario esté autentificado, que exista el paciente determinado.	
Entrada: se define un estado de evolución del paciente. Estados: menos grave, grave, critico.	
Resultado esperado: se adiciona una evolución diaria a un paciente determinado.	
Evaluación de la prueba: prueba satisfactoria.	

Tabla 28 PA Calcular índice pronóstico.

Pruebas de aceptación	
Código:4	No. HU: 4
Nombre: Calcular índice pronóstico.	
Descripción: prueba para calcular índices pronósticos.	
Condiciones de ejecución: que el usuario esté autentificado, que exista el paciente en cuestión.	
Entrada: que realiza la entrada de datos válidos de los índices pronósticos. Datos: hemoglobina, presión arterial, potasio, etc.	
Resultado esperado: se calcula y adiciona un índice pronóstico a un paciente.	

Evaluación de la prueba: prueba satisfactoria.

3.5 Resultados de las pruebas

Durante el despliegue del sistema en el Hospital Enrique Cabrera, en una primera iteración fueron reportadas 5 no conformidades, las cuales fueron corregidas en su totalidad. En una segunda iteración realizada no fueron encontradas no conformidades. A continuación se muestra el resultado de estas pruebas en la figura 12:

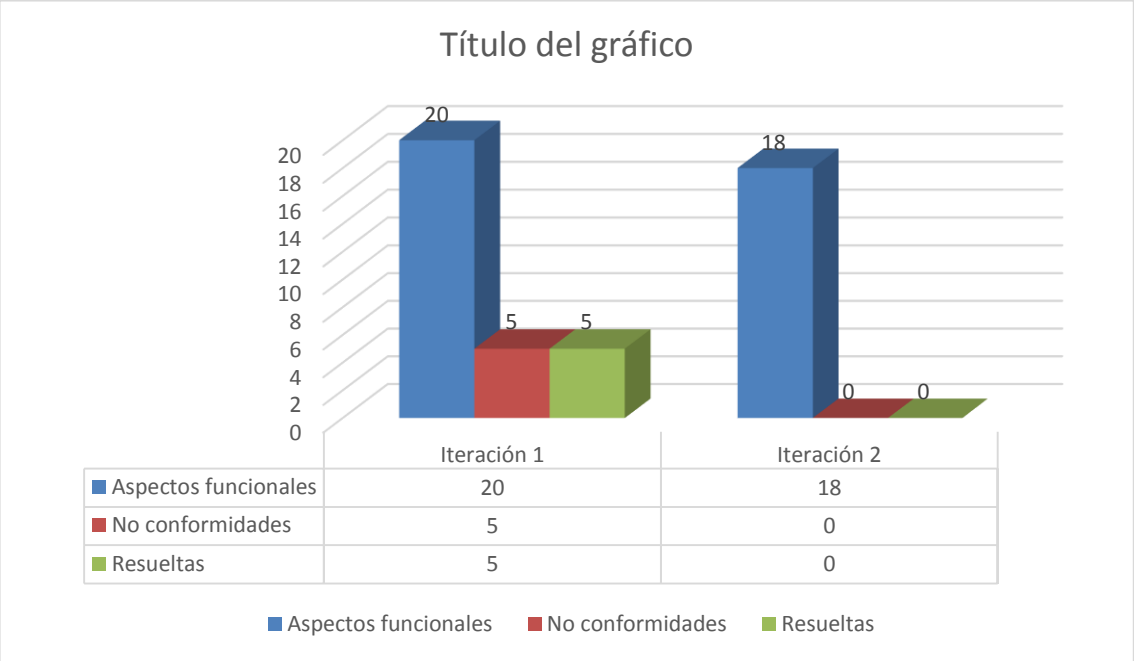

Figura 12 Resultado de las pruebas de aceptación.

Producto de los resultados satisfactorios que arrojaron las pruebas de aceptación que se le realizaron a cada uno de los aspectos funcionales definidos, queda garantizado que el sistema en general cumple con el objetivo para la cual fue diseñado

El sistema fue instalado en el nodo del Hospital Enrique Cabrera, para la realización de las pruebas, siendo ejecutado en un total de 21 PC con sistema operativo Windows, sin mostrar problemas de velocidad en las respuestas. La revisión del sistema por parte del cliente, trajo consigo el recibimiento de un aval de calidad para el sistema por parte de Leonardo Concepción Quiñones. Este aval demuestra que el mismo, está apto para ser usado en el Hospital Enrique Cabrera, y si se decide, posteriormente en el resto de los centros hospitalarios del país.

3.6 Conclusiones

El análisis de los resultados de la etapa de implementación y prueba del sistema, además de poner a prueba la aplicación con datos reales, sirvió para identificar mejoras

en el proceso de gestión de pacientes intervenidos en el área abdominal, concluyendo que:

- Las TI permitieron dar cumplimiento a las HU definidas por el cliente.
- El sistema cumple con los aspectos tanto funcionales como los no funcionales identificados para el mismo.
- Las pruebas realizadas sirvieron para identificar el grado de aceptación del cliente, y para su posterior despliegue en el Hospital Enrique Cabrera.
- Las pruebas unitarias sirvieron para elevar la calidad del código.

CONCLUSIONES GENERALES

Luego de la realización de la investigación se arriba a las siguientes conclusiones:

- El establecimiento de los fundamentos teórico–metodológico de la investigación realizada permitió conceptualizar los procesos de la gestión de la información en pacientes intervenidos quirúrgicamente en el área abdominal.
- La realización del análisis y diseño permitió la materialización del sistema de gestión de información para el apoyo al diagnóstico de pacientes intervenidos quirúrgicamente en el área abdominal.
- El sistema desarrollado para el proceso de entrada y salida de datos, permitió obtener una herramienta que se apoya en siete índices pronósticos para predecir la morbilidad y mortalidad en los pacientes intervenidos quirúrgicamente en el área abdominal, lo que permite elevar los niveles de integridad de la información, puesto que no existen sistemas con estas características, solucionando así los problemas presentados.
- Las distintas pruebas realizadas al sistema desarrollado permitió garantizar el aumento de los niveles de integridad en el proceso de entrada y salida de datos.

RECOMENDACIONES

El objetivo general trazado para el desarrollo del sistema fue alcanzado satisfactoriamente, pero se recomienda:

- La implementación de nuevas funcionalidades con la ayuda de la inteligencia artificial, como son, el análisis basado en casos y la minería de datos.
- La implementación de otro grupo de índices pronósticos (APACHE III, FMO), con el objetivo de ampliar el alcance del sistema.

BIBLIOGRAFÍA

1. **Alvarez, Miguel Angel. 2009.** DesarrolloWeb. [En línea] 2009. http://www.desarrolloweb.com/articulos/codeigniter.html.

2. **Álvarez, Miguel Angel. 2007.** Editores para PHP, Zend Studio. [En línea] 2007. http://www.adrformacion.com/cursos/php/leccion1/tutorial3.html.

3. **Alvarez, Sara. 2007.** Desarrollo Web. [En línea] 2007.

4. **Álvarez, Sara. 2007.** DesarrolloWeb. *DesarrolloWeb.* [En línea] Septiembre de 2007. http://www.desarrolloweb.com/articulos/modelo-entidad-relacion.html.

5. **Angel Alvarez, Miguel. 2001.** DesarrolloWeb. [En línea] 2001. http://www.desarrolloweb.com/articulos/541.php.

6. **—. 2003.** DesarrolloWeb. [En línea] 2003. http://www.desarrolloweb.com/articulos/1325.php.

7. **CakePHP. 2008.** CakePhP. [En línea] 2008. cakephp.org.

8. **Calero Solís, Manuel. 2003.** *Una explicación de la programación extrema (XP).* 2003.

9. **Care2x. 2002.** Care2x. [En línea] 2002. http://www.care2x.org/..

10. **Ciberaula.com. 2008.** Introducción a Apache. [En línea] 2008. http://linux.ciberaula.com/articulo/linux_apache_intro/..

11. **Cotos Yáñez, José Manuel y Taboada González, José Ángel. 2005.** *Sistemas de información medioambiental.* 2005.

12. **Domenico Fraccalvieri, Sebastiano Biondo. 2009.** ScienceDirect. [En línea] Noviembre de 2009. http://www.sciencedirect.com/science/article/pii/S0009739X09002061.

13. **eclipse.org. 2008.** Eclipse. *Eclipse.* [En línea] 2008. http://www.eclipse.org/org/.

14. **Eguiluz Perez, Javier. 2010.** *Introducción a JavaScript .* 2010.

15. **Eguiluz, Javier. 2009.** *CSS Avanzado.* 2009.

16. **ElWebMaster. 2009.** [En línea] 2009. http://www.elwebmaster.com/articulos/frameworks-php-recomendados-guia-para-principiantes.

17. **Farrera, M. A. 2000.** Sistemas de Información para Hospitales. [En línea] 2000. http://genesis.uag.mx/posgrado/revistaelect/compu/his_archivos/frame.htm.

18. **Fernández Escribano, Gerardo. 2002.** *Introducción a Extreme Programming.* 2002.

19. **Fowler, Martin. 2000.** *Metodologías Ágiles: eXtreme Programming.* 2000.

20. **Framework, Zend. 2009.** Oficial Web Site. [En línea] 2009. framework.zend.com.

21. **González Barbone, Víctor A. 2007.** *XP: Extreme Programming.* 2007.

22. **Gutierrez, J J y otros, y. 2010.** PRUEBAS DEL SISTEMA EN PROGRAMACIÓN EXTREMA. [En línea] 2010. [Citado el: 20 de Marzo de 2014.]

23. **IBM. 2009.** Rational Rose. [En línea] 2009. http://www-03.ibm.com/software/products/es/enterprise/.

24. **JELSOFT. 2008.** [En línea] 2008. http://foro.ignetwork.net/showthread.php?t=15188.

25. **Letelier, Patricio y Penadés, Maria Del Carmen. 2009.** *Métodologías ágiles para el desarrollo de software: eXtreme Programming.* Valencia : s.n., 2009.

26. **López Álvarez, JM y González Jorge, R. 2001.** Medicina Intensiva. *Medicina Intensiva.* [En línea] Febrero de 2001. [Citado el: 20 de Abril de 2014.] http://www.medintensiva.org/es/ndices-pronosticos-mortalidad-evaluacion-una/articulo/12003084/.

27. **Medicina, Universidad Autónoma de México. D. R. Facultad de. 2003.** Sistema de Información Hospitalaria. México D.F. [En línea] 2003. http://www.facmed.unam.mx/emc/computo/ssa/HIS/his.pdf.

28. **Merayo Castellano, Isidro. 2012.** Frameworks de Tests Doubles en PHP. [En línea] Febrero de 2012. http://www.programania.net/diseno-de-software/frameworks-de-tests-doubles-en-php-i/.

29. **Microsoft. 2011.** Microsoft Developer Network. [En línea] 2011. http://msdn.microsoft.com/es-es/library/ms174219.aspx.

30. **Packo. 2012.** [En línea] 2012. http://packo.wikispaces.com/Caracteristicas+de+MYSQL.

31. **PHP.net. 2010.** php.net. [En línea] 2010. http://www.php.net/. .

32. **PostgreSQL.org. 2012.** PostgreSQL. [En línea] 2012. http://postgresql-dbms.blogspot.com/p/limitaciones-puntos-de-recuperacion.html.

33. **S.A, Valen Computer. 2008.** Valen Computer S.A. [En línea] 2008. [Citado el: http://www.valen.es/cas/hospitales.html de Enero de 2014.]

34. **Sebastián, Juán. 2010.** Comusoft. [En línea] Noviembre de 2010. http://www.comusoft.com/modelo-vista-controlador-definicion-y-caracteristicas.

35. **SunMicrosystem. 2008.** SunMicrosystem. [En línea] 2008. www.visual-paradigm.com..

36. **SUSHRT. 2006.** SUSHRT. [En línea] 2006. http://www.cdac.in/html/his/sushrut.asp..

37. **Symfony.es. 2010.** Symfony.es. *Symfony.es.* [En línea] 2010. http://symfony.es/que-es-symfony.

38. **System, Sparx. 2008.** [En línea] 2008. http://www.sparxsystems.com/products/ea/.

39. **Vega, John Freddy y Van Der Henst, Christian. 2011.** Maestros del Web. *Guia de HTML5.* [En línea] 2011. http://mlw.io/guia-html5/.

40. **Wheeler, Veronika. 2008.** Linux en Espoñol. [En línea] 2008. http://www.linuxespanol.com/viewtopic.php?p=96480&sid=c1e2f61c2c9f25c5ca0e492a6c8.

ANEXOS

Anexo 1: Historias de usuarios

Tabla 29 HU Modificar paciente.

Historia de usuario	
Número: 5	Usuario: Usuario
Nombre: Modificar paciente	
Prioridad en el negocio: alta	Riesgo de desarrollo: medio
Puntos de estimación: 1	Iteración: 1
Programador responsable: Frank Delfín Reyes	
Descripción: El sistema debe brindar la posibilidad de modificar los datos de ingreso correspondientes a un paciente.	

Tabla 30 HU Listar paciente.

Historia de usuario	
Número: 6	Usuario: Usuario
Nombre: Listar paciente	
Prioridad en el negocio: baja	Riesgo de desarrollo: bajo
Puntos de estimación: 1	Iteración: 1
Programador responsable: Frank Delfín Reyes	
Descripción: El sistema debe listar todos los pacientes insertados, mostrando sus datos básicos (Número de Historia clínica, nombre, primer apellido, sexo, raza), además de la opción de abrir un expediente seleccionado.	

Tabla 31 HU Eliminar políticas de antibióticos.

Historia de usuario	
Número: 8	Usuario: Usuario
Nombre: Eliminar políticas de antibióticos	
Prioridad en el negocio: media	Riesgo de desarrollo: bajo
Puntos de estimación: 1	Iteración: 2
Programador responsable: Lázaro E. Millares Bermúdez	
Descripción: El sistema debe brindar la posibilidad de eliminar políticas de antibióticos asociadas a un paciente.	

Tabla 32 Listar políticas de antibióticos.

Historia de usuario	
Número: 9	Usuario: Usuario
Nombre: Listar políticas de antibióticos	
Prioridad en el negocio: media	Riesgo de desarrollo: bajo
Puntos de estimación: 1	Iteración: 2
Programador responsable: Lázaro E. Millares Bermúdez	
Descripción: El sistema debe brindar la posibilidad de listar todas las políticas de antibióticos aplicadas a un paciente, mostrando datos como el nombre del antibiótico, la fecha de esta y la fecha de fin en caso de estar terminada, la dosis y las acciones que se pueden realizar sobre esta (terminar, eliminar).	

Tabla 33 Adicionar antibióticos.

Historia de usuario	
Número: 10	Usuario: Usuario
Nombre: Adicionar antibióticos	
Prioridad en el negocio: alta	Riesgo de desarrollo: bajo
Puntos de estimación: 1	Iteración:2
Programador responsable: Frank Delfín Reyes	
Descripción: El sistema debe brindar la posibilidad de adicionar un antibiótico, insertando su nombre.	

Tabla 34 Modificar antibióticos.

Historia de usuario	
Número: 11	Usuario: Usuario
Nombre: Modificar antibióticos	
Prioridad en el negocio: alta	Riesgo de desarrollo: bajo
Puntos de estimación: 1	Iteración:2
Programador responsable: Frank Delfín Reyes	
Descripción: El sistema debe brindar la posibilidad de modificar un antibiótico insertado.	

Tabla 35 Eliminar antibióticos.

Historia de usuario	
Número: 12	Usuario: Usuario
Nombre: Eliminar antibióticos	
Prioridad en el negocio: alta	Riesgo de desarrollo: bajo
Puntos de estimación: 1	Iteración:2
Programador responsable: Lázaro E. Millares Bermúdez	
Descripción: El sistema debe brindar la posibilidad de eliminar un antibiótico insertado.	

Tabla 36 Listar antibióticos.

Historia de usuario	
Número: 13	Usuario: Usuario
Nombre: Listar antibióticos	
Prioridad en el negocio: alta	Riesgo de desarrollo: bajo
Puntos de estimación: 1	Iteración:2
Programador responsable: Lázaro E. Millares Bermúdez	
Descripción: El sistema debe listar todos los antibióticos insertados.	

Tabla 37 Eliminar evolución del paciente.

Historia de usuario	
Número: 14	Usuario: Usuario
Nombre: Eliminar evolución del paciente	
Prioridad en el negocio: alta	Riesgo de desarrollo: bajo
Puntos de estimación: 2	Iteración: 1
Programador responsable: Frank Delfín Reyes	
Descripción: El sistema deberá permitir eliminar la evolución de un paciente.	

Tabla 38 Adicionar índices pronósticos.

Historia de usuario	
Número: 15	Usuario: Usuario
Nombre: Adicionar índices pronósticos	
Prioridad en el negocio: alta	Riesgo de desarrollo: bajo
Puntos de estimación: 1	Iteración: 1
Programador responsable: Lázaro E. Millares Bermúdez	
Descripción: El sistema deberá permitir adicionar los índices pronósticos.	

Tabla 39 Modificar índices pronósticos.

Historia de usuario	
Número: 16	Usuario: Usuario
Nombre: Modificar índices pronósticos	
Prioridad en el negocio: alta	Riesgo de desarrollo: bajo
Puntos de estimación: 1	Iteración: 2
Programador responsable: Lázaro E. Millares Bermúdez	
Descripción: El sistema deberá permitir modificar los datos insertados de las variables de los índices pronósticos.	

Tabla 40 Eliminar índices pronósticos.

Historia de usuario	
Número: 17	Usuario: Usuario
Nombre: Eliminar índices pronósticos	
Prioridad en el negocio: alta	Riesgo de desarrollo: bajo
Puntos de estimación: 1	Iteración: 2
Programador responsable: Frank Delfín Reyes	
Descripción: El sistema deberá permitir eliminar los índices pronósticos.	

Tabla 41 Listar índices pronósticos.

Historia de usuario	
Número: 18	Usuario: Usuario
Nombre: Listar índices pronósticos	
Prioridad en el negocio: alta	Riesgo de desarrollo: bajo
Puntos de estimación: 1	Iteración: 1
Programador responsable: Lázaro E. Millares Bermúdez	
Descripción: El sistema debe listar los diferentes índices pronósticos.	

Tabla 42 Realizar búsqueda de pacientes.

Historia de usuario	
Número: 19	Usuario:
Nombre: Realizar búsqueda de pacientes	
Prioridad en el negocio: alta	Riesgo de desarrollo: medio
Puntos de estimación: 1	Iteración: 2
Programador responsable: Frank Delfín Reyes	
Descripción: El sistema deberá brindar la posibilidad de buscar pacientes introduciendo un valor que deberá comprobarse con los campos diagnósticos, con el número de historia clínica, y con el nombre y apellidos de todos los pacientes.	

Tabla 43 Listar historial.

Historia de usuario	
Número: 20	Usuario: Usuario
Nombre: Listar historial	
Prioridad en el negocio: alta	Riesgo de desarrollo: medio
Puntos de estimación: 1	Iteración: 2
Programador responsable: Frank Delfín Reyes	
Descripción: El sistema debe listar todos los pacientes dados de alta, mostrando sus datos básicos (Número de Historia clínica, nombre, primer apellido, sexo, raza), además de la opción de abrir un expediente seleccionado, y en caso de que este haya sido dado de alta vivo este puede reingresarse.	

Anexo 2: Tareas de ingeniería.

Tabla 44 TI Modificar paciente.

Tarea de ingeniería	
No. De la tarea: 6	No. De la HU:2
Nombre de la terea: Modificar paciente	
Tipo de tarea: Desarrollo	Puntos estimados: 0.1
Fecha inicio:1/15/2014	Fecha fin:1/22/2014
Programador responsable: Lázaro E. Millares Bermúdez	
Descripción: Implementar las funcionalidades necesarias para modificar los datos de un paciente ya insertado en el sistema. Bundles: pacienteBundle Entidades: paciente	

Tabla 45 TI Listar paciente.

Tarea de ingeniería	
No. De la tarea: 7	No. De la HU:3
Nombre de la terea: Listar paciente	
Tipo de tarea: Desarrollo	Puntos estimados: 0.1
Fecha inicio:1/15/2014	Fecha fin:1/22/2014
Programador responsable: Lázaro E. Millares Bermúdez	

Descripción: Desarrollar una interfaz que permita listar todos los pacientes insertados en el sistema. Bundles: pacienteBundle. Entidades: paciente.

Tabla 46 Modificar políticas de antibióticos.

Tarea de ingeniería	
No. De la tarea: 8	No. De la HU:5
Nombre de la terea: Modificar políticas de antibióticos	
Tipo de tarea: Desarrollo	Puntos estimados: 0.2
Fecha inicio:1/22/2014	Fecha fin:1/27/2014
Programador responsable: Frank Delfín Reyes	
Descripción: El especialista o el administrador introducirán los datos de la historia clínica del paciente. De estos grupos de datos existen campos obligatorios y campos opcionales.	

Tabla 47 TI Eliminar política de antibióticos.

Tarea de ingeniería	
No. De la tarea: 9	No. De la HU:6
Nombre de la terea: Eliminar política de antibióticos	
Tipo de tarea: Desarrollo	Puntos estimados: 0.1
Fecha inicio:1/22/2014	Fecha fin:1/27/2014
Programador responsable: Lázaro E. Millares Bermúdez	
Descripción: Implementar las funcionalidades necesarias para eliminar una política de antibióticos asignada a un paciente. Bundles: pacienteBundle. Entidades: paciente, políticaantibióticos, antibiótico.	

Tabla 48 TI Listar política de antibióticos.

Tarea de ingeniería	
No. De la tarea: 10	No. De la HU:7
Nombre de la terea: Listar política de antibióticos	
Tipo de tarea: Desarrollo	Puntos estimados: 0.1

Fecha inicio:2/2/2014	Fecha fin:2/8/2014
Programador responsable: Frank Delfín Reyes	
Descripción: Desarrollar una interfaz que permita lista las políticas de antibióticos aplicadas a un paciente seleccionado. Bundles: pacienteBundle. Entidades: paciente, políticaantibióticos, antibiótico.	

Tabla 49 TI Modificar antibióticos.

Tarea de ingeniería	
No. De la tarea: 11	No. De la HU:9
Nombre de la terea: Modificar antibióticos	
Tipo de tarea: Desarrollo	Puntos estimados: 0.1
Fecha inicio:2/9/2014	Fecha fin:2/13/2014
Programador responsable: Lázaro E. Millares Bermúdez	
Descripción: Desarrollar las funcionalidades necesarias para modificar un antibiótico existente en el sistema. Bundles: pacienteBundle. Entidades: antibióticos.	

Tabla 50 TI Eliminar antibióticos.

Tarea de ingeniería	
No. De la tarea: 12	No. De la HU:10
Nombre de la terea: Eliminar antibióticos	
Tipo de tarea: Desarrollo	Puntos estimados: 0.1
Fecha inicio:2/15/2014	Fecha fin:2/20/2014
Programador responsable: Lázaro E. Millares Bermúdez	
Descripción: Desarrollar las funcionalidades necesarias para eliminar un antibiótico existente en el sistema. Bundles: pacienteBundle. Entidades: antibióticos.	

Tabla 51 TI Listar antibióticos.

Tarea de ingeniería	
No. De la tarea: 13	No. De la HU:11
Nombre de la terea: Listar antibióticos	
Tipo de tarea: Desarrollo	Puntos estimados: 0.1
Fecha inicio:2/15/2014	Fecha fin:2/22/2014
Programador responsable: Frank Delfín Reyes	
Descripción: Desarrollar una interfaz para listar todos los antibióticos existentes en el sistema Bundles: pacienteBundle. Entidades: antibióticos.	

Tabla 52 TI Adicionar índices pronósticos.

Tarea de ingeniería	
No. De la tarea: 14	No. De la HU:15
Nombre de la terea: Adicionar índices pronósticos	
Tipo de tarea: Desarrollo	Puntos estimados: 0.1
Fecha inicio:2/21/2014	Fecha fin:2/27/2014
Programador responsable: Lázaro E. Millares Bermúdez	
Descripción: Desarrollar una interfaz que permita insertar los datos de los índices pronósticos. Bundles: pacienteBundle, predictorBundle Entidades: paciente, predictor.	

Tabla 53 TI Modificar índices pronósticos.

Tarea de ingeniería	
No. De la tarea: 15	No. De la HU:16
Nombre de la terea: Modificar índices pronósticos	
Tipo de tarea: Desarrollo	Puntos estimados: 0.1
Fecha inicio:3/1/2014	Fecha fin:3/5/2014
Programador responsable: Lázaro E. Millares Bermúdez	

Descripción: Implementar las funcionalidades necesarias para modificar un índice pronostico. Bundles: pacienteBundle, predictorBundle Entidades: paciente, predictor.

Tabla 54 Eliminar índices pronósticos.

Tarea de ingeniería	
No. De la tarea: 16	No. De la HU:17
Nombre de la terea: Eliminar índices pronósticos	
Tipo de tarea: Desarrollo	Puntos estimados: 0.1
Fecha inicio:3/6/2014	Fecha fin:3/10/2014
Programador responsable: Frank Delfín Reyes	
Descripción: Implementar las funcionalidades necesarias para eliminar un índice pronostico. Bundles: pacienteBundle, predictorBundle Entidades: paciente, predictor.	

Tabla 55 TI Listar índices pronósticos.

Tarea de ingeniería	
No. De la tarea: 17	No. De la HU:18
Nombre de la terea: Listar índices pronósticos	
Tipo de tarea: Desarrollo	Puntos estimados: 0.1
Fecha inicio:3/15/2014	Fecha fin:3/19/2014
Programador responsable: Frank Delfín Reyes	
Descripción: Descripción: Desarrollar una interfaz para listar todos los datos de un índice pronóstico existente en el sistema. Bundles: pacienteBundle. Entidades: antibióticos.	

Tabla 56 TI Realizar búsqueda de pacientes.

Tarea de ingeniería	
No. De la tarea:18	No. De la HU:19

Nombre de la terea: Realizar búsqueda de pacientes	
Tipo de tarea: Desarrollo	Puntos estimados: 0.1
Fecha inicio:3/15/2014	Fecha fin:3/25/2014
Programador responsable: Lázaro E. Millares Bermúdez	
Descripción: Implementar las funciones necesarias para buscar pacientes en correspondencia con sus campos. Bundles: pacienteBundle. Entidades: paciente	

Tabla 57 Listar historial.

Tarea de ingeniería	
No. De la tarea:19	No. De la HU:20
Nombre de la terea: Listar historial	
Tipo de tarea: Desarrollo	Puntos estimados: 0.1
Fecha inicio:3/25/2014	Fecha fin:3/30/2014
Programador responsable: Frank Delfín Reyes	
Descripción: Descripción: Desarrollar una interfaz que permita listar todos los pacientes dados de alta. Bundles: pacienteBundle. Entidades: paciente.	

Tabla 58 Eliminar evolución del paciente.

Tarea de ingeniería	
No. De la tarea:20	No. De la HU:13
Nombre de la terea: Eliminar evolución del paciente	
Tipo de tarea: Desarrollo	Puntos estimados: 0.2
Fecha inicio:4/1/2014	Fecha fin:4/8/2014
Programador responsable: Lázaro E. Millares Bermúdez	
Descripción: El especialista o el administrador introducirán los datos de la historia clínica del paciente. De estos grupos de datos existen campos obligatorios y campos opcionales.	

Anexo 3: Pruebas de aceptación.

Tabla 59 PU Modificar paciente.

Pruebas de aceptación	
Código: 5	No. HU: 2
Nombre: Modificar paciente	
Descripción: Prueba para modificar un paciente.	
Condiciones de ejecución: Que el usuario esté autenticado. Que el paciente exista.	
Entrada: Se intenta la entrada de datos válidos y no duplicados en el sistema. Los datos son: número de historia clínica, nombre, primer apellido, segundo apellido, dirección, edad, sexo, raza, numero de carnet de identidad.	
Resultado esperado: Se crea el paciente en el sistema.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 60 PU Listar paciente.

Pruebas de aceptación	
Código: 6	No. HU: 3
Nombre: Listar paciente	
Descripción: Prueba para listar los pacientes insertados en el sistema y no dados de alta.	
Condiciones de ejecución: Que el usuario esté autenticado. Que el existan pacientes.	
Entrada: Ninguna.	
Resultado esperado: Se deben listar todos los pacientes insertados en el sistema y no dados de alta.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 61 PU Modificar política de antibióticos.

Pruebas de aceptación	
Código: 7	No. HU: 5
Nombre: Modificar política de antibióticos	
Descripción: Prueba para modificar una política de antibióticos	
Condiciones de ejecución: Que el usuario esté autenticado	
Entrada: Se intenta la entrada de datos válidos y no duplicados en el sistema	
Resultado esperado: Se modifique la política de antibióticos con los datos nuevos	
Evaluación de la prueba: Satisfactoria	

Tabla 62 PU Eliminar política de antibióticos.

Pruebas de aceptación	
Código: 8	No. HU: 6
Nombre: Eliminar política de antibióticos	
Descripción: Prueba para eliminar una política de antibiótico.	
Condiciones de ejecución: Que el usuario esté autenticado. Que el paciente exista. Que tenga asignada al menos una política de antibióticos.	
Entrada: Ninguna.	
Resultado esperado: Se debe eliminar la política de antibióticos seleccionada.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 63 PU Listar política de antibióticos.

Pruebas de aceptación	
Código: 9	No. HU: 7
Nombre: Listar política de antibióticos	
Descripción: Prueba para listar las políticas de antibióticos.	
Condiciones de ejecución: Que el usuario esté autenticado. Que el paciente exista. Que tenga asignada al menos una política de antibióticos.	
Entrada: Ninguna.	
Resultado esperado: Se deben listar todas las políticas de antibióticos asignadas a un paciente.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 64 PU Adicionar antibióticos.

Pruebas de aceptación	
Código: 10	No. HU: 8
Nombre: Adicionar antibióticos	
Descripción: Prueba para adicionar antibióticos al sistema.	
Condiciones de ejecución: Que el usuario esté autenticado.	
Entrada: Se intenta la entrada de datos válidos y no duplicados en el sistema. Los datos son: Nombre.	
Resultado esperado: Se crea el antibiótico en el sistema.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 65 PU Modificar antibióticos.

Pruebas de aceptación	
Código: 11	No. HU: 9
Nombre: Modificar antibióticos	
Descripción: Prueba para modificar un antibiótico.	
Condiciones de ejecución: Que el usuario esté autenticado. Que el antibiótico exista.	
Entrada: Se intenta la entrada de datos válidos y no duplicados en el sistema. Los datos son: Nombre.	
Resultado esperado: Se modifica el antibiótico seleccionado.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 66 PU Eliminar antibióticos.

Pruebas de aceptación	
Código: 12	No. HU: 10
Nombre: Eliminar antibióticos	
Descripción: Prueba para eliminar un antibiótico.	
Condiciones de ejecución: Que el usuario esté autenticado. Que el antibiótico exista.	
Entrada: Ninguna.	
Resultado esperado: Se elimine el antibiótico seleccionado.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 67 PU Listar antibióticos.

Pruebas de aceptación	
Código: 13	No. HU: 11
Nombre: Listar antibióticos	
Descripción: Prueba para listar los antibióticos.	
Condiciones de ejecución: Que el usuario esté autenticado. Que exista al menos un antibiótico.	
Entrada: Ninguna.	
Resultado esperado: Se listen los antibióticos.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 68 PU Eliminar evolucion del paciente.

Pruebas de aceptación	
Código: 14	No. HU: 13

Nombre: Eliminar evolución del paciente
Descripción: Prueba para eliminar la evolución de un paciente
Condiciones de ejecución: Que el usuario esté autenticado
Entrada: Ninguna
Resultado esperado: Se elimine la evolución del paciente
Evaluación de la prueba: Satisfactoria

Tabla 69 PU Adicionar índices pronósticos.

Pruebas de aceptación	
Código: 15	No. HU: 15
Nombre: Adicionar índices pronósticos	
Descripción: Prueba para adicionar un índice pronóstico.	
Condiciones de ejecución: Que el usuario esté autenticado. Que no existan índices pronósticos del tipo seleccionado adicionados en el día.	
Entrada: Se intenta la entrada de datos válidos y no duplicados en el sistema.	
Resultado esperado: Se añade un índice pronóstico.	
Evaluación de la prueba: prueba satisfactoria.	

Tabla 70 PU Modificar índices pronósticos.

Pruebas de aceptación	
Código: 16	No. HU: 16
Nombre: Modificar índices pronósticos	
Descripción: Prueba para modificar un índice pronóstico.	
Condiciones de ejecución: Que el usuario esté autenticado. Que exista el índice pronóstico.	
Entrada: Se intenta la entrada de datos válidos y no duplicados en el sistema.	
Resultado esperado: Se modifica un índice pronóstico.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 71 PU Eliminar índices pronósticos.

Pruebas de aceptación	
Código: 17	No. HU: 17
Nombre: Eliminar índices pronósticos	
Descripción: Prueba para eliminar un índice pronóstico.	

Condiciones de ejecución: Que el usuario esté autenticado. Que exista el índice pronóstico.
Entrada: Ninguna.
Resultado esperado: Se elimine el índice pronóstico seleccionado.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 72 PU Listar índices pronósticos.

Pruebas de aceptación	
Código: 18	No. HU: 18
Nombre: Listar índices pronósticos	
Descripción: Prueba para listar los índices pronósticos.	
Condiciones de ejecución: Que el usuario esté autenticado. Que exista al menos un índice pronóstico.	
Entrada: Ninguna.	
Resultado esperado: Se liste los índices pronósticos.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 73 PU Realizar búsqueda pacientes.

Pruebas de aceptación	
Código: 19	No. HU: 19
Nombre: Realizar búsqueda pacientes	
Descripción: prueba para buscar pacientes.	
Condiciones de ejecución: Que el usuario esté autenticado. Que exista al menos un paciente.	
Entrada: Datos de un paciente. Los datos pueden ser: Nombre, diagnósticos, número	
Resultado esperado: Se listan los pacientes que coincidan.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 74 PU Listar historial.

Pruebas de aceptación	
Código: 20	No. HU: 20
Nombre: Listar historial	
Descripción: prueba para listar los pacientes del historial (dados de alta)	
Condiciones de ejecución: Que el usuario esté autenticado. Que exista al menos un paciente dado de alta.	

Entrada: Ninguna.
Resultado esperado: Lista de los pacientes dados de alta.
Evaluación de la prueba: Prueba satisfactoria.